

VIRGINIA POLYTECHNIC INST AND STATE UNIV BLACKSBURG --ETC F/G 17/7
STAR PATTERN RECOGNITION AND SPACECRAFT ATTITUDE DETERMINATION. (U)
MAY 81 T E STRIKWERDA, J L JUNKINS DAAK70-78-C-0038

ETL-0260

NL

42A
101500

10809

LEVEL

4

ETI .0260

AD A103806

**Star pattern recognition and
spacecraft attitude determination**

Thomas E. Strikwerda

John L. Junkins

**Virginia Polytechnic Institute
and State University
Blacksburg, Virginia 24061**

DTIC

SELECTE

SEP 2 1981

H

MAY 1981

DTIC FILE COPY

**APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED**

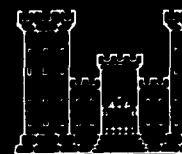
Prepared for

U.S. ARMY CORPS OF ENGINEERS

ENGINEER TOPOGRAPHIC LABORATORIES

FORT BELVOIR, VIRGINIA 22060

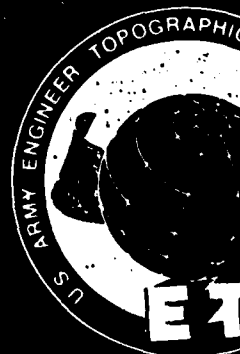
81 9 02 016



E

T

L



Destroy this report when no longer needed.
Do not return it to the originator.

The findings in this report are not to be construed as an official
Department of the Army position unless so designated by other
authorized documents.

The citation in this report of trade names of commercially available
products does not constitute official endorsement or approval of the
use of such products.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

17 REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM	
1. REPORT NUMBER ETL 0260	2. GOVT ACCESSION NO. AD-A103806	3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) STAR PATTERN RECOGNITION AND SPACECRAFT ATTITUDE DETERMINATION		5. TYPE OF REPORT & PERIOD COVERED Contract Report	
7. AUTHOR(s) Thomas E. Strikwerda John L. Junkins		6. PERFORMING ORG. REPORT NUMBER	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Engineering Science & Mechanics Department Virginia Polytechnic Institute & State University Blacksburg, Virginia 24061		8. CONTRACT OR GRANT NUMBER(s) DAAK70-78-C-0038	
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Engineer Topographic Laboratories Fort Belvoir, Virginia 22060		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE May 1981	
		13. NUMBER OF PAGES 196	
		15. SECURITY CLASS. (of this report) Unclassified	
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE	
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited			
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)			
18. SUPPLEMENTARY NOTES			
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) spacecraft, attitude, orientation, navigation, stellar camera, pattern recognition			
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) A method for real-time on-board spacecraft attitude determination is presented. The method is suitable for processing CCD or CID stellar camera and rate gyro output to determine orientation to better than five arc seconds. The system is self-calibrating; uncertain parameters such as interlock angles and gyro biases can be included in the estimation algo- rithms. The system is implemented in a microcomputer system which con- clusively establishes that the system is compatible with on-board computation constraints.			

DD FORM 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

PREFACE

This document is the final report of Contract DAAK70-78-C-0038 for the U.S. Army Engineer Topographic Laboratories, Fort Belvoir, Virginia.

The authors appreciate the capable guidance of Mr. L. A. Gambino, Director of the Computer Science Laboratory (USAETL), who served as Technical Monitor for this effort.

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
By Distribution	
By	
Distribution/	
Availability Codes	
Avail and/or	
Dist	Special
A	

SUMMARY

The primary results of the research efforts are the following:

1. Development of an approach for real time on-board estimation of spacecraft orientation with sub five arc-second precision.
2. Detailed formulation of an efficient and reliable star pattern recognition strategy appropriate for use with charged-coupled-device (CCD) array-type star sensors.
3. Formulation of a motion integration/Kalman filter algorithm to integrate gyro measured angular rates and (by sequential processing of the discrete orientation information available from the star sensing, identification, and attitude determination process) provide optimal real time estimates of spacecraft orientation and angular velocity.
4. Development of truth models to generate realistic input data for the star pattern recognition and Kalman filter strategies.
5. Formulation of algorithms using Euler parameters to define orientation.
6. Implementation and validation of the approach in a laboratory microcomputer - the objective being to assess the problems associated with a real-time, on-board version of this system.

These results are discussed in detail herein. This report is organized in such a fashion that the key features and results of the work are discussed in the main body of the text; the more involved and technical details are documented in the ten appendices.

TABLE OF CONTENTS

	<u>Page</u>
Preface	i
Summary	ii
1.0 Introduction and System Overview	1
2.0 Orientation Parameters and Coordinate Frames	10
3.0 Process B	16
4.0 Process C	22
5.0 Truth Model and Simulation Tests	26
6.0 Conclusions	49
7.0 References	51
<u>Appendix</u>	
1 CCD Star Tracker	52
2 CCD Instrument Response and Stellar Magnitude Conversion	54
3 Star Data Base and Mission Catalog Creation	61
4 Inter-Star Cosine Calculations	65
5 Stellar Aberration	67
6 Least-Squares Correction Techniques	71
7 Orientation of the Vehicle Frame	76
8 Riccati Equation Covariance Propagation	81
9 Rate Gyro Readout Data Generation	87
10 Software Documentation	89

1.0 Introduction

This document reports the findings of a three year research project to develop a method for on-board satellite attitude determination. We believe this method can achieve sub-five-arcsecond accuracy when applied to data obtained with a new general purpose star tracker (typical of several existing configurations).

The primary motivation for the research is the exploitation of recently developed light sensitive Charge-Coupled-Devices (CCD) arrays, placed in the focal plane of a tracker lens, to act as a "film" for imaging starlight. Satellite attitude can be determined by identifying the stars detected by the CCD. The star image data, output from the CCD, can be either telemetered to ground for later analysis or, as described in this report, analyzed on-board (via computers configured in parallel) to determine satellite attitude autonomously in near real-time.

The basic system we propose consists of 2 or 3 CCD star trackers and 3 microcomputers, each with a dedicated function. The function of each of the 4 sub-systems is outlined below, with reference to Figures 1.1, 1.2, and 1.3.

1.1 System Overview

(1) CCD Star Sensors and Associated Electronics

Although the development of CCD sensors and trackers is not part of this research, there are several CCD star tracker designs proposed by various organizations involved in hardware development. The purpose of our work has been exploitation of the CCD star tracker technology; we have chosen a particular set of parameters

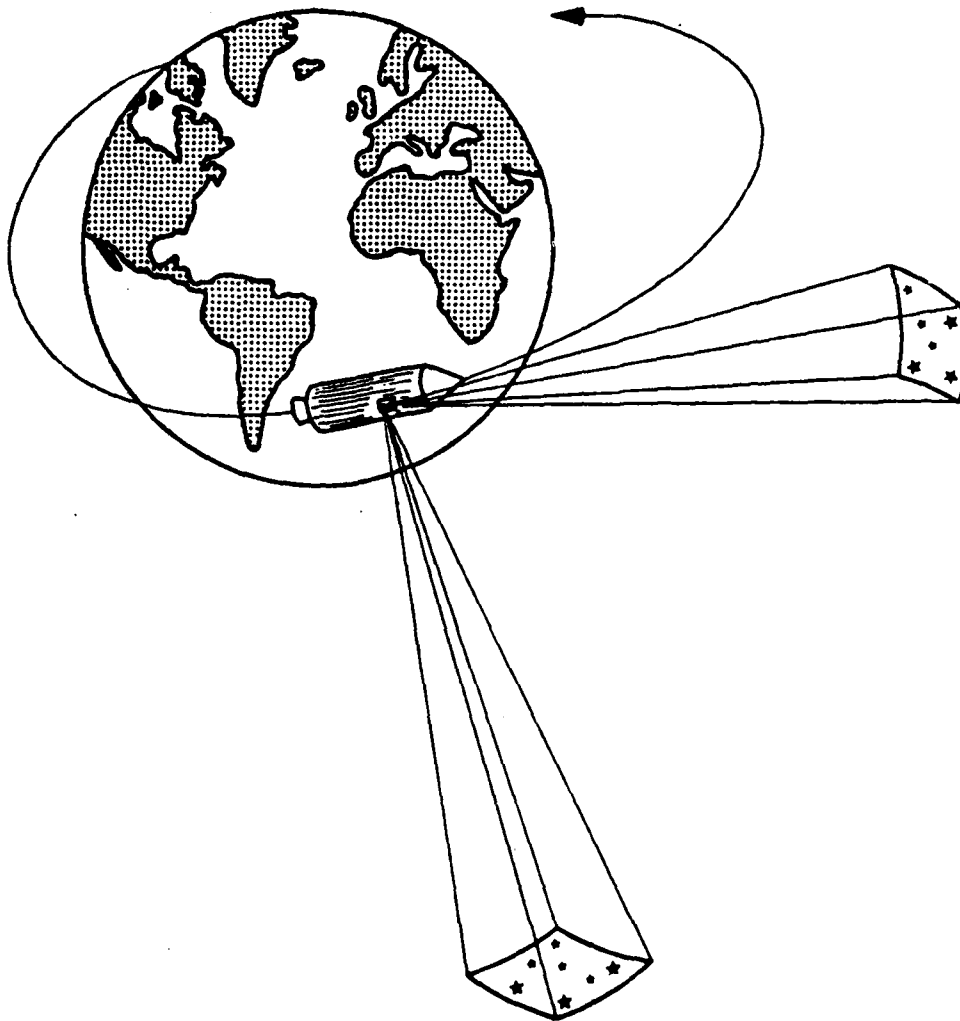


Figure 1.1 UVASTAR An electro-optical/software system capable of real time readout of digitized star coordinates, and ultimately, autonomous, near-real time star pattern recognition and attitude determination.

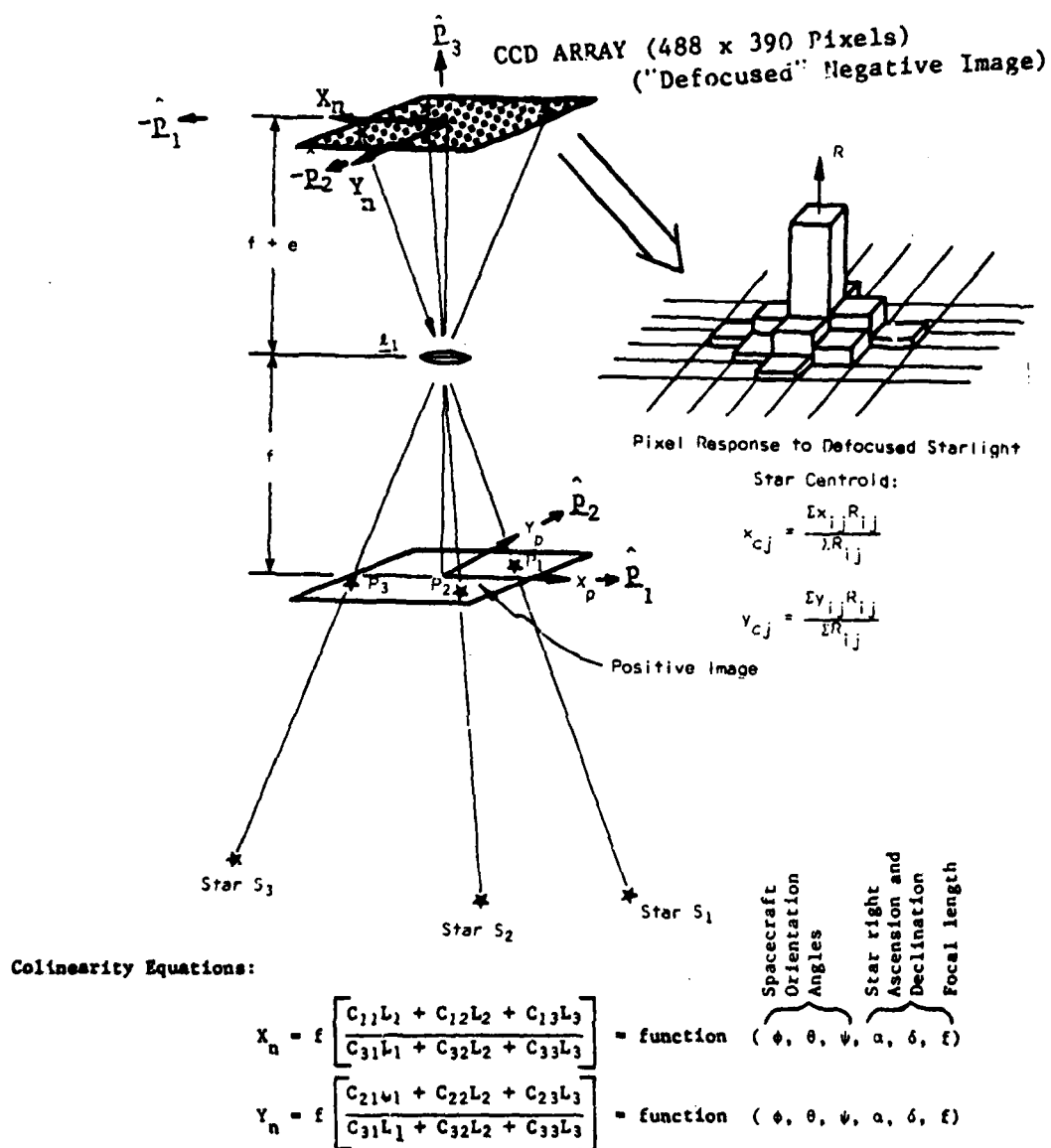
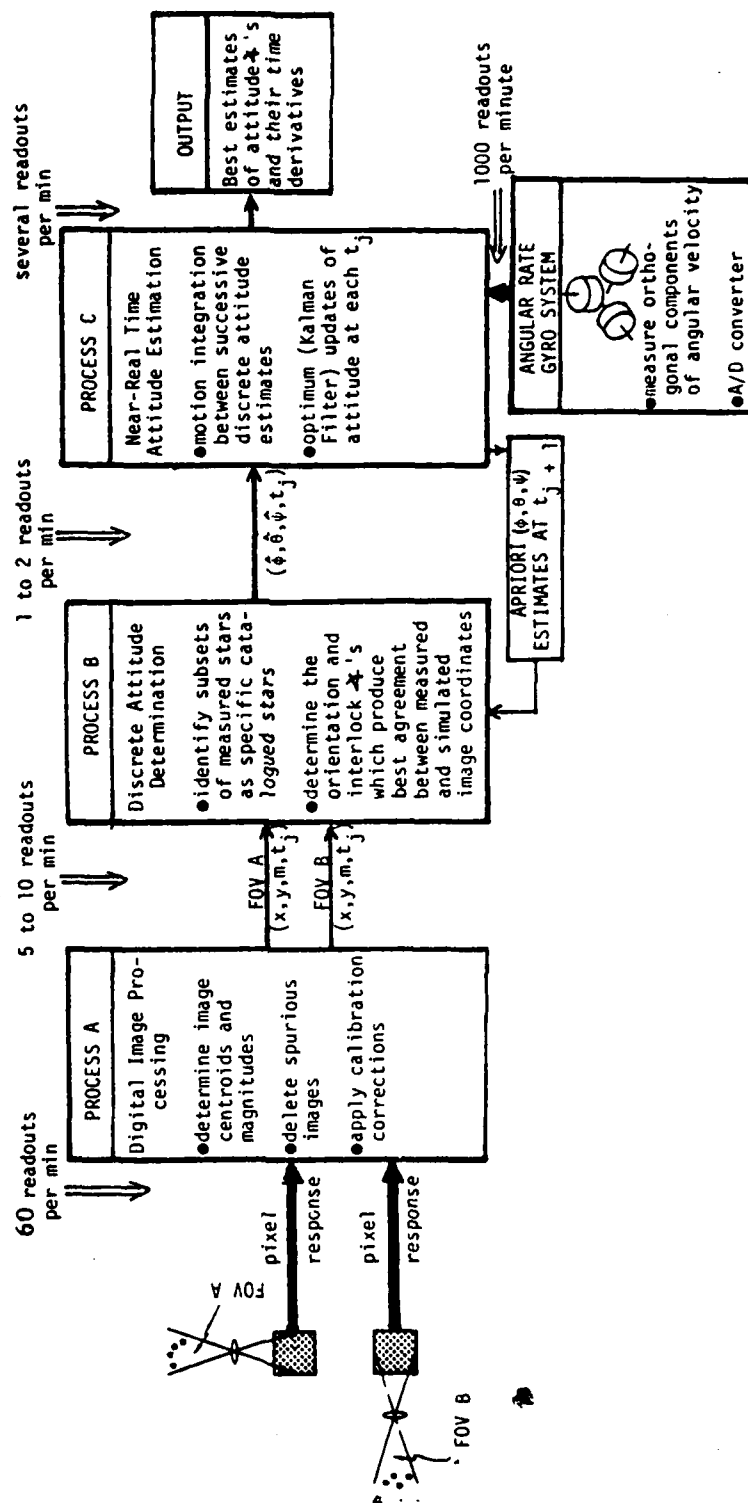


Figure 1.2 Formation of Image on the CCD Array.

Figure 1.3
STAR PATTERN RECOGNITION/SPACECRAFT ATTITUDE ESTIMATION



but it should be kept in mind that these are nominal, achievable values without further CCD/star tracker technology advances. Each of the two (or three) trackers is identical and their boresights are assumed separated by the nominal interlock angle of 90° . We assume a lens focal length of 70 mm and a CCD array size such that a $7^\circ \times 9^\circ$ field of view (FOV) is imaged onto the array. The arrays are assumed to be Fairchild's 11.4 mm x 8.8 mm matrix consisting of 488×380 silicon pixels, each pixel accurately imbedded (to 1 part in 10,000) in a microcircuit chip. Starlight is defocused slightly on the CCD in order to spread typical images over 9 to 16 neighboring pixels. This permits accurate "centroiding" of the image to determine image coordinates accurate to about 10% of a pixel (10% is a conservative estimate). The processing of a data frame consists of a rapid sequential readout of the voltage response of all pixels and an analog to digital (A/D) conversion only of certain pixels (based upon response above an analog threshold level or prior selection). The scans of each field of view are controlled by a common clock and are assumed to represent 2 (or 3) frames (1 from each sensor) taken at the same instant. This assumption is valid for all but very rapidly spinning satellites, since the CCDs can be scanned 10 times per second. (Refer to Appendix 1 for more star tracker information.)

(2) Microcomputer A

Program *Process A* is performed by a Microcomputer A with either one computer per sensor or sequential treatment of data for 2 or 3 sensors. Again, Process A is not part of this research program but since the functions to be performed are straightforward, we simply replace Process A by calculating synthetic output data

whose availability is clock controiled. Process A takes as input data the digitized pixel voltages and pixel coordinates for up to 10 stars in each FOV and the associated time. Image centroids are calculated for each image and corrections for lens distortion and other known error sources are applied; a relative magnitude or intensity is also calculated. As output, Process A delivers the focal plane coordinates for each star image. Since Process A calculations for one data frame can be performed in near real time and many times faster than the attitude can be determined, it may be possible and desirable to perform additional editing of the star data. For example, images with rapidly varying image intensity from frame to frame could be eliminated or images whose successive positions are inconsistent with the overall motion caused by vehicle motion (such as images of space debris) could be deleted immediately from consideration (failure to detect and delete all spurious images does not prove fatal, but does slow the pattern recognition logic of Process B). Process A would be expected to output image coordinate and magnitude data at the rate of about 5 frames per sensor each minute and simply overwrite old data. The microcomputer of Process A is considered as an integral part of the star tracker itself, making it a "smart sensor".

Since Process A controls the scan of the CCD and its electronics, it is possible to track only those stars desired (those whose pixel response lies within specified bounds). Thus, even though the CCD array contains thousands of pixels, only a small fraction of their response values need be subjected to A/D conversion and stored at any one time. It is this data compaction feature, along with the

high dimensional stability of CCD arrays that make them so attractive for this application. Also significant is the high speed readout of the CCD which allows one to assume, for most cases, that the star images visible in a given frame have been imaged simultaneously. Therefore, stellar resection (geometric) methods can be used for attitude determination and the vehicle motion can be ignored for analysis of a single frame of data.

(3) Microcomputer B

Data from Process A (and Process C) are analyzed by program *Process B*; again by means of a dedicated microcomputer. As input, Process B accepts:

- star image coordinates and magnitude data; one set per FOV (from Process A);
- a-priori attitude estimates and covariance, (from Process C); and
- a-priori estimates and covariance of interlock angles between the sensors image planes (from previous analysis of Process B data).

The sequence of calculations/logical decisions divides into two primary functions:

- identify measured stars in each FOV as specific stars contained in an on-board star catalog (containing, in the general case, the direction cosines and instrument magnitudes of the 5000 brightest stars) and
- determine the spacecraft orientation and field of view interlock angles which cause the simulated images of identified catalog stars to overlay the corresponding measured images in a least-squares sense.

These two tasks will be discussed in detail in Section 3 and appendices. The expected output rate for Process B is two or more attitude updates per minute of elapsed time. The old attitude and covariance, output to Process C, are overwritten by each new attitude and covariance.

(4) Microcomputer C

The attitude determined by Process B for a discrete time is further processed by program *Process C* in microcomputer C. Input to this program consists of the attitude and covariance from Process B and A/D converted gyro rate measurements of angular velocity. The kinematic differential equations governing the spacecraft attitude are integrated forward from the attitude determined from the previous pass through Process C (using the gyro rate measurements). This yields an attitude estimate at the time associated with the next set of image coordinates from Process A. After Process B has determined the discrete attitude it is combined with the integrated attitude in a Kalman Filter calculation to give a best estimate of attitude at the time associated with the star tracker data. Further forward integration gives an estimated attitude and covariance at real-time.

1.2 Focus of This Study

Our primary tasks were to develop the algorithms for Process B (star pattern recognition and attitude determination), and Process C (state integration and Kalman filter routines). This, of course, required some study of CCD arrays and star tracker design (Process A). A governing principle was that this system be suitable for a general purpose satellite; that is, we did not design it with a particular mission in

mind. We have required a slowly rotating satellite, however, in order to insure that star images do not cause streaks in the star camera and that our rate integration be valid (i.e., the vehicle not undergo rapid maneuvers).

The algorithms we devised can run on a large memory (64,000 bytes), general purpose microcomputer. To demonstrate this, we have programed the algorithms on a Hewlett-Packard 9845S microcomputer equipped with a high level BASIC interpreter language package. Although the processing time with this language is significantly slower than a compiler type of system or machine code program its use permitted programming ease, which was essential for development work. Our tests show the present system will produce updated attitude estimates every 60 seconds (with rate integrated attitude available several times per second) in a steady state mode; when the programs are implemented in a form suitable for a satellite computer they should execute much faster.

We have organized this report to include most of the detail and mathematical developments in appendices in order to keep the body descriptive and concise. Section 2 discusses the coordinate frames and orientation variables used in this study. Processes B and C are described in Sections 3 and 4, respectively. We discuss our truth model and simulation tests of our algorithms in Section 5 and present conclusions in Section 6. The reader is referred to references 1-3 for a discussion of intermediate results of this project.

2.0 Orientation Parameters and Coordinate Frames

In order to describe the orientation of a spacecraft we need to specify some coordinate frame fixed in the vehicle and another fixed in inertial space. In addition, we need a parameter set to describe the relative orientation of these two frames.

Euler angles provide an easily understood description of relative orientation of two frames. The three angles specify a sequence of rotations about three successive coordinate axes of a rotated frame. However, although they are descriptive, Euler angles are not very suitable for our purposes for several reasons. Any of the twelve possible rotation sequences possesses two singularities. In addition, the differential equations describing the rotational motion of a vehicle involve trigonometric nonlinearities when expressed in terms of Euler angles. The same is true of the least-squares equations used in the star pattern recognition algorithms. Extensive use of trigonometric functions will significantly increase the computation time.

2.1 Euler Parameters

These problems have been circumvented by using a set of four variables called Euler parameters instead of Euler angles. Euler parameters have the advantages that (1) they do not have a geometric singularity, (2) they rigorously satisfy linear differential equations, and (3) no evaluation of trigonometric functions need be done in any application discussed herein. One disadvantage is the four parameters must sum-square to unity; we have found methods to include this constraint in our estimation algorithms.

Euler parameters ($\beta_0, \beta_1, \beta_2, \beta_3$) can be interpreted geometrically in terms of Euler's theorem: A completely general angular displacement

of a rigid body can be accomplished by a single rotation (the principal angle, ϕ) about a line (the principal line, $\hat{\ell}$) which is fixed relative to both arbitrary body-fixed axes $\{\hat{b}\}$ and reference axes $\{\hat{n}\}$. If $\{\hat{n}\}$ is initially coincident with $\{\hat{b}\}$, then the direction cosines (ℓ_1, ℓ_2, ℓ_3) of $\hat{\ell}$ with respect to $\{\hat{n}\}$ and $\{\hat{b}\}$ are identical.

The Euler parameters are then related to the principal rotation parameters as follows:

$$\begin{aligned}\beta_0 &= \cos \phi/2 \\ \beta_i &= \ell_i \sin \phi/2, \quad i = 1, 2, 3.\end{aligned}\tag{2.1}$$

Note that Euler parameters satisfy the constraint:

$$\sum_{i=0}^3 \beta_i^2 = 1.\tag{2.2}$$

The rotation matrix $[C]$ characterizing the relationship between a body fixed frame $\{\hat{b}\}$ and a reference frame $\{\hat{n}\}$ by: $\{\hat{b}\} = [C]\{\hat{n}\}$ can be written in terms of Euler parameters as:

$$[C] = \begin{bmatrix} \beta_0^2 + \beta_1^2 - \beta_2^2 - \beta_3^2 & 2(\beta_1\beta_2 + \beta_0\beta_3) & 2(\beta_1\beta_3 - \beta_0\beta_2) \\ 2(\beta_1\beta_2 - \beta_0\beta_3) & \beta_0^2 - \beta_1^2 + \beta_2^2 - \beta_3^2 & 2(\beta_2\beta_3 + \beta_0\beta_1) \\ 2(\beta_1\beta_3 + \beta_0\beta_2) & 2(\beta_2\beta_3 - \beta_0\beta_1) & \beta_0^2 - \beta_1^2 - \beta_2^2 + \beta_3^2 \end{bmatrix}\tag{2.3}$$

2.2 Coordinate Frames

For convenience we have used several coordinate frames for Process B and C algorithms. They are: the inertial frame, N, the gyroscope frame, G, the vehicle frame, V, and two camera frames, A and B (Table 2.1).

The inertial frame is our primary reference frame and is defined, essentially, by star positions. The locations of all the stars in the

Table 2.1

COORDINATE FRAMES

Inertial Frame (N):	Primary reference frame. Used for star positions and vehicle velocity components.
Gyroscope Frame (G):	Defined by orientation of three orthogonal gyroscopes. Rotation rate of the vehicle is measured in this frame.
Camera "A" Frame (A):	Defined by orientation of camera boresight and focal plane.
Camera "B" Frame (B):	Defined by orientation of camera boresight and focal plane.
Vehicle Frame (V):	Defined by boresight unit vectors of the "A" and "B" frames. Orientation of this frame with respect to the "N" frame is determined by Process B.

RELATIONSHIPS BETWEEN FRAMES

- G - N: Changes as vehicle rotates.
- V - N: Changes as vehicle rotates.
- V - G: Assume this varies slowly with time. Gyro bias terms compensate for small, slow variations.
- B - A: Assume this varies slowly with time. Interlock parameters are monitored by Process B.

onboard catalog are specified in this frame, as are the vehicle velocity components (used for aberration corrections). The gyro frame is defined by the axes of three orthogonal gyroscopes, fixed in the vehicle. Each gyroscope gives a measure of the vehicle rotation rate about that axis (these are the rates integrated by Process C). Our simulation studies have been configured for a nominally earth pointing spacecraft. Accordingly, we have specified that the unit vectors $\{g_i\}$ along the gyro axes be oriented such that g_3 is along the radius vector, g_2 is perpendicular to the orbit plane and then $g_1 = g_2 \times g_3$ (nominally along the velocity vector).

The two camera frames, A and B, are assumed fixed to the vehicle and, therefore, maintain a fixed orientation with respect to the gyro frame. We have specified that \underline{a}_3 and \underline{b}_3 coincide with the camera boresights and point 45° from the direction of vehicle motion, above and below the orbit plane. Unit vectors \underline{a}_1 and \underline{b}_1 lie along the x axis of the CCD of each camera and lie in the orbit plane while \underline{a}_2 and \underline{b}_2 form the y axis of each CCD.

The V frame has been defined by the boresight vectors, \underline{a}_3 and \underline{b}_3 (see Figure 2.2):

$$\begin{aligned}\underline{v}_1 &= (\underline{a}_3 + \underline{b}_3) / |\underline{a}_3 + \underline{b}_3| \\ \underline{v}_2 &= \underline{v}_3 \times \underline{v}_1 \\ \underline{v}_3 &= (\underline{a}_3 \times \underline{b}_3) / |\underline{a}_3 \times \underline{b}_3| .\end{aligned}\tag{2.1}$$

Both Processes B and C have been formulated to employ the Euler parameters which orient this vehicle frame with respect to the inertial frame.

There are several advantages to this definition of the V frame. First, the boresight vector of each frame is well determined compared

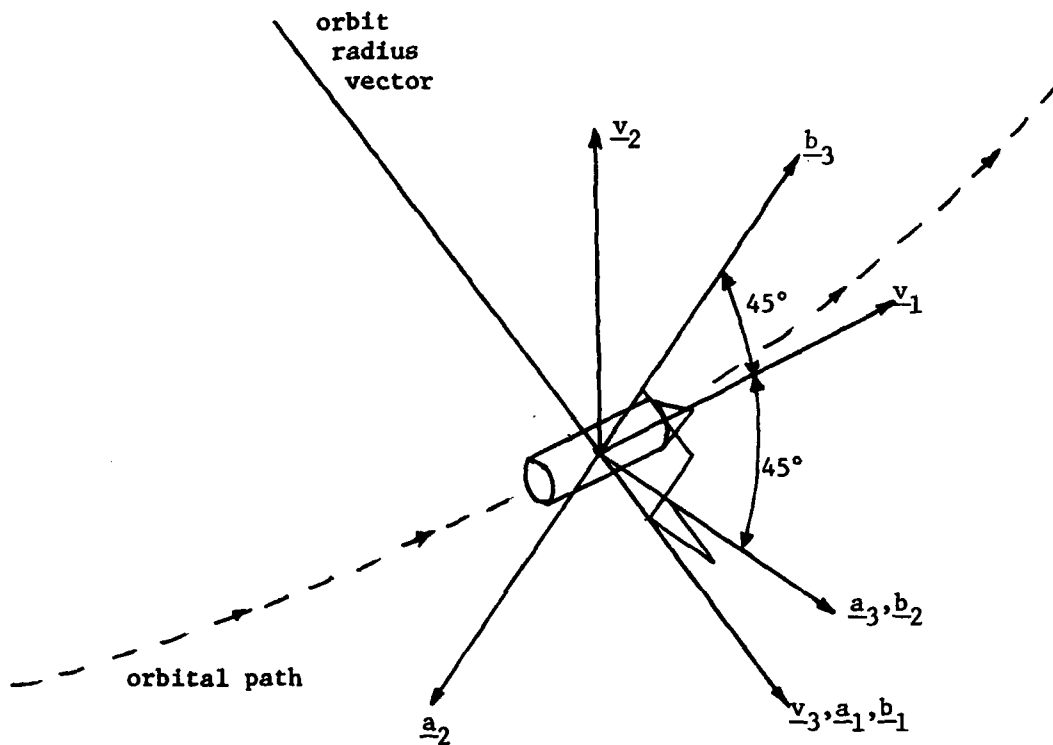


Figure 2.1. Relationship of the vehicle frame to FOV(A) and FOV(B)

with the rotation about the boresight vector. Thus, the V frame orientation is not affected by the poorly known quantities. Second, by using this definition we weight frames A and B equally.

Although frames G, A and B are nominally fixed with respect to the vehicle, in reality these *interlock* relative orientations will vary due to thermal cycling, vehicle vibrations, etc. Therefore, we have included techniques in Processes B and C, to be discussed in later sections, to monitor and/or partially correct for these interlock variation effects. A by-product is the attractive feature that the system becomes fully self-calibrating.

3.0 Process B

Attitude determination by processing image coordinates obtained from Process A depends upon the ability to describe mathematically the location of a star image on the CCD image plane, given its direction in space and the orientation of the star tracker. This mapping, a function of the Euler parameters discussed in Section 2, is described by the stellar colinearity equations which, for frame A, have the form:

$$x = f \frac{L_1 AN_{11} + L_2 AN_{12} + L_3 AN_{13}}{L_1 AN_{31} + L_2 AN_{32} + L_3 AN_{33}} + x_0$$

$$y = f \frac{L_1 AN_{21} + L_2 AN_{22} + L_3 AN_{23}}{L_1 AN_{31} + L_2 AN_{32} + L_3 AN_{33}} + y_0$$
(3.1)

where f = lens focal length, assumed to be constant,

AN_{ij} = elements of the coordinate frame rotation matrix AN ,
in turn a function of Euler parameters, and

L_i = star direction cosines for the particular star as
measured in the N frame

(x_0, y_0) = principal point offsets.

If there are several stars in a single field of view (FOV) we seek to minimize the sum of the squares of the residuals between measured star images and predicted coordinates for the same stars. This is accomplished by adjusting the Euler parameters, which orient the star tracker frame, using a least-square differential correction scheme.

Before outlining the least-square procedures, we describe the process of equating particular catalog stars with measured stars. To start Process B we need an estimate of the camera orientation. This can be provided by either the results of a previous pass through Processes B and C or from some indirect method such as horizon sensors. This

estimate is needed to acquire a "sub-catalog" from the mission catalog and must be sufficiently accurate so that the subcatalog contains the measured stars.

3.1 Star Catalog

As part of our work on Process B, we have converted the visual magnitude of over 5,000 stars to a standard infra-red (I) magnitude. For simplicity, we assumed the instrument magnitude is identical to the I magnitude (which could be arranged by using an I filter). In specific applications instrument magnitude would probably be based upon laboratory calibration. We have not precessed the star positions nor corrected for proper motion since these tasks would best await an actual flight test of the system. Details of magnitude conversions are discussed in Appendix 2.

In addition, we developed a star catalog format for easy access. The celestial sphere is divided into cells or segments in an orderly pattern so that any cell can be accessed easily to obtain the positions of stars contained within. It is important to keep in mind that our catalog segmentation and access logic were designed for a general mission. Simplified catalogs could be designed for specific missions. See Appendix 3 for more details on the cell structure and access logic.

3.2 Star Pairing

Associating catalog and measured stars begins by sorting catalog stars by angular distance off the apriori estimated FOV boresight direction. By computing the vector dot product of each star with the boresight vector we have a suitable measure of angular distance and can sort stars according to this parameter (and thereby avoid repetitive

angle calculations from inverse trigonometric functions). The next step is to compute and store in a table, the cosine of the interstar angle for all possible pairs of measured stars. We then pair catalog stars, beginning with stars nearest the estimated boresight, compute the cosine of the interstar angle, and then compare this value with each value for the measured pairs (refer to Appendix 4). This process is repeated until either a match is found to within some tolerance or the list of catalog stars is exhausted. In the latter case we start over with fresh data from Process A and a new estimate of orientation. However, if a match is found, we tentatively assume the catalog pair is the same as the measured pair. We are now ready to adjust the estimated orientation parameters via least-squares correction to get the two projected catalog stars to overlay the two measured stars. Because of the relatively high probability of finding an invalid star pair match, attitude confirmation requires additional star matches, as discussed below. We must also account for the effect of stellar aberration on the star direction cosines (refer to Appendix 5).

3.3 Least-Squares Correction

The non-linear relationship between the Euler parameters and star image coordinates requires an iterative least-squares correction procedure to find the best estimate of vehicle orientation. Basically, at each iteration we require the Euler parameter corrections to minimize:

$$(\Delta X - A \Delta \beta)^T W (\Delta X - A \Delta \beta) \quad (3.2)$$

where ΔX is a column vector of x and y coordinate residuals between the measured and predicted images (using the current values for the orientation variables), A is a matrix of partial derivatives of star positions

(the stellar colinearity equations) with respect to current Euler parameters, $\Delta\beta$ is the correction vector to be added to the current parameters and W is a weight matrix. The derivation of this equation is found in Appendix 6.

Since the Euler parameters must satisfy a constraint equation, it is necessary to guarantee that the corrected Euler parameters also satisfy this constraint. If we express the constraint equation as

$$\beta^T \beta = 1, \quad (3.3)$$

then, after correcting the parameters, the corrections $\Delta\beta$ must satisfy:

$$(\beta + \Delta\beta)^T (\beta + \Delta\beta) = 1. \quad (3.4)$$

Expanding to first order we have:

$$\beta^T \beta + 2\beta^T \Delta\beta = 1 + \text{residual} \quad (3.5)$$

and by writing this as

$$(1 - \beta^T \beta) - 2\beta^T \Delta\beta = \text{residual} \quad (3.6)$$

we can append $1 - \beta^T \beta$ to the ΔX vector, $2\beta^T$ to the A matrix and $\Delta\beta$ is again the correction vector. In solving Eq. (3.2) we assign a large weight to this constraint equation in order to insure that it is satisfied (i.e., the residual will be essentially zero).

After the vector of Euler parameters has been found by iteration, it is necessary to confirm whether or not the catalog pair is indeed the measured pair (i.e., whether we have the correct orientation). Each catalog star is mathematically projected onto the focal plane and tested to see if it lies near a measured star. A match of three or more stars is considered a positive outcome; a match of only two stars (most likely the initial pair) or fewer constitutes failure and we continue with star

pair matching to find another pair. In the present software version we accept up to 5 catalog stars which match measured stars in one FOV.

The star pair matching and confirmation calculations described above are performed separately for each FOV. If the outcome for each FOV is positive, we have up to 5 measured stars from each FOV with their corresponding catalog positions. All of these stars are used to correct the orientation again and, in addition, to correct the Euler parameters defining the interlock relationship between the two FOV. We again minimize:

$$(\Delta X - A \Delta \beta)^T W (\Delta X - A \Delta \beta).$$

Now, ΔX contains the residuals for all images, the A matrix contains partial derivatives with respect to both the Euler parameters orienting the vehicle frame and those orienting frame B with respect to A, and $\Delta \beta$ contains corrections to these same Euler parameters. As before, we append two constraint equations, one for each set of Euler parameters, to the matrix equation. (Refer to Appendix 7 for details of this procedure).

This method yields an accurate β_{VN} vector compared with β_{BA} . The β_{VN} describe the orientation of the vehicle frame which, in turn, is determined by the FOV boresight vectors, both usually well determined. On the other hand, the β_{BA} are effected by the relatively poorer determination of the roll angle about the boresight vector of each FOV. Therefore, we have found it desirable to further process β_{BA} . We assume the true β_{BA} vary slowly (due to such things as thermal cycling) and write:

$$\dot{\beta}_{BA} = 0 \tag{3.7}$$

and then combine the apriori or predicted values of β_{BA} (obtained from a previous analysis) with the calculated values of β_{BA} obtained via

least-squares. The two vectors of β_{BA} are combined using a discrete Kalman filter (see Appendix 6 for details). This method can be used, with proper tuning, to monitor the interlock variations and give the system "memory" of past interlock determinations.

We note that the least-squares method for two FOV and the Kalman filter calculations involve considerable mathematics, such as matrix multiplication and matrix inversion, which adversely affects execution time. However, it is important, we feel, to provide the option to calibrate (as often as necessary) the interlocks between camera frames. By monitoring these variations we can make the system self-calibrating and can tolerate modest lack of mechanical stability in the various interlocks.

4.0 Process C

Process C software has two primary functions: (1) integrate the kinematic differential equations describing the satellite motion over a short time interval in order to provide Process B with a new attitude estimate, and (2) combine this integrated orientation with the orientation determined by least-squares in Process B. The second function is performed via a discrete Kalman filter to yield an optimal estimate of the orientation at a particular time.

4.1 Kinematic Equations

The differential equations describing the kinematics of a rotating coordinate frame with respect to a fixed frame, expressed in terms of Euler parameters, are:

$$\{\dot{\beta}\} = \begin{Bmatrix} \dot{\beta}_0 \\ \dot{\beta}_1 \\ \dot{\beta}_2 \\ \dot{\beta}_3 \end{Bmatrix} = \frac{1}{2} \begin{bmatrix} -\beta_1 & -\beta_2 & \beta_3 \\ \beta_0 & -\beta_3 & \beta_2 \\ \beta_3 & \beta_0 & -\beta_1 \\ \beta_2 & \beta_1 & \beta_0 \end{bmatrix} \begin{Bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{Bmatrix} = [\beta]\{\omega\} \quad (4.1)$$

$$= \frac{1}{2} \begin{bmatrix} 0 & -\omega_1 & -\omega_2 & -\omega_3 \\ \omega_1 & 0 & \omega_3 & -\omega_2 \\ \omega_2 & -\omega_3 & 0 & \omega_1 \\ \omega_3 & \omega_2 & -\omega_1 & 0 \end{bmatrix} \begin{Bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{Bmatrix} = [\omega]\{\beta\} \quad (4.2)$$

where $\{\beta\}$ are the Euler parameters orienting the frame and $\{\omega\}$ are the gyro rates measured in that frame, along the 3 orthogonal axes.

In our model we prefer to use the Euler parameters orienting the vehicle frame, V, with respect to the inertial frame, N. Therefore, the

gyro rates, measured in the G frame, must be transformed into the V frame via matrix VG, which we assume to be constant. In addition, the measured gyro rates, $\{\tilde{\omega}\}$, contain noise terms and other effects such as errors due to nonorthogonality of the gyroscopes, variations in the V-G interlocks, and gravity or magnetic effects. We account for these effects, to first order, by absorbing all except gyro noise into *bias* terms, $\{b\}$, one per axis. The equations become (using letter subscripts to denote the appropriate coordinate frame relationships):

$$\begin{aligned}\{\dot{\beta}_{VN}\} &= [\beta_{VN}][VG]\{\tilde{\omega}_{GN} - b_{GN}\} \\ &= [\tilde{\omega}_{VN}]\{\beta_{VN}\} - [\beta_{VN}][VG]\{b_{GN}\}\end{aligned}$$

where $\{\tilde{\omega}_{GN}\} = \{\omega_{GN}\}(\text{true}) + \{b_{GN}\} + \{\text{noise}\}$. The gyro biases are assumed to be slowly varying; this allows us to write:

$$\{\dot{b}_{GN}\} \begin{Bmatrix} \dot{b}_1 \\ \dot{b}_2 \\ \dot{b}_3 \end{Bmatrix} = 0$$

which is valid over short time intervals. The set of seven differential equations (for the four Euler parameters and three biases) can be integrated via Runge-Kutta methods to yield a new orientation estimate for Process B.

4.2 Kalman Filter Equations

The two estimates of vehicle attitude, one from integration of the kinematic equations and the second from Process B attitude estimation, are combined to give a best estimate of the attitude. We have adopted the discrete Kalman filter equations for a linear system:

$$\hat{X}_{k+1}(k+1) = \bar{X}_k(k+1) + K(k+1)\{\tilde{Y}(k+1) - \bar{Y}(k+1)\}$$

$$P_k(k+1) = P_k(k) + \int_{t_k}^{t_{k+1}} \dot{P} dt \quad (4.5a-d)$$

$$K(k+1) = P_k(k+1)H^T(k+1) \left\{ L_{V_{k+1}V_{k+1}} + H(k+1)P_k(k+1)H^T(k+1) \right\}^{-1}$$

$$P_{k+1}(k+1) = [I - K(k+1)H(k+1)]P_k(k+1)$$

where

$\hat{X}_{k+1}(k+1)$ = optimal estimate of the state X at time t_{k+1} based on $k+1$ data sets,

$\bar{X}_k(k+1)$ = state X at time t_{k+1} based on k data sets, and calculated from forward integration of kinematic equations,

$K(k+1)$ = Kalman gain matrix for time t_{k+1} ,

$\tilde{Y}(k+1) = \begin{Bmatrix} \beta_{VN} \\ \delta_{GN} \end{Bmatrix}$ = Values of β_{VN} from Process B and bias values from previous iteration,

$\bar{Y}(k+1) = \begin{Bmatrix} \beta_{VN} \\ \delta_{GN} \end{Bmatrix}$ = Values of β_{VN} from integration and bias values from previous iteration,

$P_i(j)$ = 7×7 covariance matrix at time t_j based on i data sets,

$H(k+1) = \left. \frac{\partial Y}{\partial X} \right|_{t_{k+1}} = I$ (for our case),

$L_{V_{k+1}V_{k+1}}$ = covariance matrix associated with measurement of the state (or observations) from Process B (the upper left 4×4) and covariance for the biases (the lower right 3×3), and

$\int_{t_k}^{t_{k+1}} \dot{P} dt$ = integration of the matrix Riccati equation for covariance propagation (see Appendix 8).

Notice that we have included the biases as observables in our Kalman filter equations. By choosing the appropriate covariance values in matrix L and P , we can control the corrections to the biases. We have done this because we hypothesize that the biases vary slowly--or at least the effects which we are most interested in monitoring vary slowly. Our simulation tests indicate that with this formulation we can follow the bias terms added to the rate gyro data and absorb variations in the interlock matrix VG into the bias terms with little degradation in the optimal state estimate.

5.0 Truth Model and Simulation Tests

The algorithms for Process B and C were tested by processing data produced by a simulation program ("truth model") and then comparing the results with the "true" model. Each test consisted of processing a series of 29 data frames separated by 30 seconds of satellite motion. The most important input data are the image coordinates and intensities of the stars in each field of view from Process A and the most important output data are the calculated orientation from Processes B and C. The simulation program was written to include variations in several important parameters such as Euler parameters describing the relative orientation of the two camera frames, Euler parameters for the rotation from the gyro to vehicle frames, and gyro bias terms. To illustrate the performance of our algorithms for this report each series included the parameter variations of the previous model plus only one additional parameter variation.

5.1 Simulation Program

We first describe briefly the creation of simulated data. The first step is to choose an appropriate satellite orbit, specified by its semimajor axis, orbital period and inclination. To facilitate the calculation of satellite position and velocity, we make use of Herrick's two body solution (see Ref. 6, p. 155). To use this method we specify the initial position and velocity components, expressed in the inertial frame, and the associated time. All later positions and velocities can be determined by specifying the desired time and solving several equations. This same method is used to determine the earth's position and velocity at each time step. Velocity data are needed to calculate the aberration of starlight which affects the apparent star directions.

All of our tests have assumed a circular satellite orbit and a nominally earth pointing vehicle. To reflect this choice we initially orient the gyroscope frame so that the g_2 axis is perpendicular to the orbit plane and the primary vehicle rotation is about that axis. The g_3 axis is initially along the orbit radius vector, \underline{r} , and \underline{g}_1 is given by $\underline{g}_2 \times \underline{g}_3$. Since our primary orientation variables are β_{VN} , as discussed in Section 2, we obtain their initial values as follows: specify the initial values for β_{VG} and calculate the rotation matrix VG , then use the gyro unit vectors $\{\underline{g}\}$ to fill matrix GN and calculate $VN = VG \cdot GN$; β_{VN} can then be recovered from VN . All subsequent values of β_{VN} are obtained by integrating the kinematic differential equations forward in time. The gyroscope rate history, needed for the integration, is given for the G frame; therefore, the rates are rotated into the V frame by matrix VG , a function of β_{VG} , which can be either constant or time varying. See Appendix 9 for details of rate gyro data simulation.

At each time step we calculate the VN matrix from β_{VN} . Matrix BA is computed from β_{BA} (again, constant or time varying parameters) and from BA we compute AV (see Appendix 7). The last row of $AN = AV \cdot VN$ is the FOV(A) camera boresight unit vector, needed to access the star catalog for a subcatalog of stars. After adding the effects of aberration, the stars are projected onto the CCD image plane via the stellar colinearity equations. Stars seen by the second camera are obtained in the same manner, after first computing $BN = BA \cdot AN$ to get the boresight unit vector.

The image coordinates obtained by the above methods are assumed to represent the "true" state. In an actual system Process A will not,

of course, produce the true image coordinates. We have assumed that the centroiding of an image can be performed to an accuracy of 10% of a pixel (1-sigma error) and that systematic errors such as image distortion can be accounted for and removed. Therefore, we perturb the true image coordinates with Gaussian noise.

Various data are stored on tape or disk for later analysis by Process B and C. Space is left at the end of each record (one record per frame) for data computed by Processes B and C; these are later analyzed for accuracy and displayed.

5.2 Simulation Tests

A set of eight models was used to test our algorithms. All models followed the same orbital path and rotation history. Of the 29 data frames, each consisting of image coordinates in a pair of FOV and separated by 30 seconds of flight time, only once does a FOV contain 2 stars (the case at 8 minutes from the start). In that case, the least-squares solution used only the stars from one FOV; the orientation errors for this case are relatively large in all models. For display purposes, we have plotted the root-mean-square of the angular errors between the calculated and true vehicle frame (using the 1-2-3 Euler angle set). There are actually three calculated frames: the result of Process B least-squares, the integrated state, and the optimal estimate from the Kalman filter. Each series started with an estimate about 2 degrees in error. Thus, several frames must be processed for the system to reach steady state.

No noise or parameter variations were included in the first model in order to verify that the software could indeed recover the

true state (Figure 5.1). Gaussian noise added to the rate gyro data ($1 \text{ sigma} = 1 \text{ arcsecond/second}$) causes the integrated state of our second model to deviate from the true state (Figure 5.2). It is evident that with this level of noise, the state could be integrated several minutes, at least, before the accumulated error would place the estimated state too far from the true state. Thus, Process C provides adequate backup for failures of Process B. Notice also that the optimal estimate nearly matches the Process B state; this is due to the significantly smaller covariance associated with the Process B result.

Our third test included noise in the image coordinates ($1 \text{ sigma} = 0.0034 \text{ mm}$) corresponding to approximately 10 arcsecond error in determining a star's direction (lens focal length = 70 mm). Once again the optimal estimate is nearly the Process B result (Figure 5.3). This and all following simulation models show that the most important factor affecting the vehicle attitude determination accuracy is the accuracy with which individual star centroids can be determined. A reduction of centroid errors will produce a proportional reduction in orientation errors. Since the least-squares result nearly matches the optimal estimate, improving the star tracker performance will yield the most improvement in attitude estimation. Our choice of 10% pixel centroiding error for each star, yielding about 5 arcsecond vehicle pointing error, is considered conservative. Indications are that 5% pixel error can be obtained routinely, with perhaps even smaller errors for brighter stars (considerable research is presently under way to determine optimal "tuning" of the sensor and centroiding process-clearly an appropriate scale factor can be applied to our results to reflect other centroiding error models).

The fourth model included variations in the Euler parameters describing the interlock between star tracker frames. As expected, this does not seriously degrade the orientation of the vehicle (V) frame since its orientation is defined by the boresight unit vectors (Figure 5.4a). It will be recalled that Process B algorithms estimate these interlock parameters. Figures 5.4b-d display the estimates obtained for the three interlock angles. Each figure shows the deviations, from the nominal interlock angle, of the true angle, the angle calculated from least-squares and the best estimate of the interlock angle. This series used a value 5 arcseconds for the variance of process noise matrix in the Kalman filter calculations. The fifth series was identical to the forth series except we used a value of 10 arcseconds. Results of Figure 5.5a indicate little effect on the vehicle frame orientation while Figures 5.5b-d show improved interlock recovery compared with Figure 5.4b-d. The value used for the process noise should be influenced by the size of the expected variations in interlocks. A strict value (small noise) prohibits the algorithm from following a true variation while a large value leads to large fluctuations in the interlocks and no meaningful self-calibration.

The next several simulations concern Process C performance. First we added a time varying bias term to each gyro axis in order to test how well Process C algorithms recover and follow each bias. Figure 5.6a indicates the biases degrade the integrated state only slightly once the biases have been recovered (after several minutes). Figure 5.6b shows the true and calculated bias values. By choosing a different value for the bias variance (see Section 4) we can control the fluctuations in the recovered biases. To demonstrate this, in our seventh

series we increased the variance square root from 0.5 to 1.0 arcsecond/second. Figure 5.7a shows no effect on the vehicle frame determination while Figure 5.7b indicates a faster bias recovery but somewhat larger bias fluctuations compared with Figure 5.6b.

Our final simulation test included the effects of time-varying and off-set Euler parameters, β_{VG} , describing the relationship between the vehicle and gyro frames. Process C algorithms assume this relationship is fixed (in our case rotation matrix VG is the identity matrix) so any deviation will appear, over the short interval, as simply an additional bias term in the gyroscopes. Thus, we see little effect in the V frame errors (Figure 5.8a) but notice the recovered bias values are displaced somewhat from their previous tracks (Figure 5.8b, variance square-root is 0.5 arcsecond/second). We assume that other slowly varying or constant effects will be accounted for in like manner.

The choice for the bias variance should be influenced by the expected variations in the gyro biases as well as an estimate of variations in other elements. As with the FOV interlock weight, a strict value (small variance) restricts the tracking of a true variation while a liberal value negates the self-calibrating nature of the algorithms.

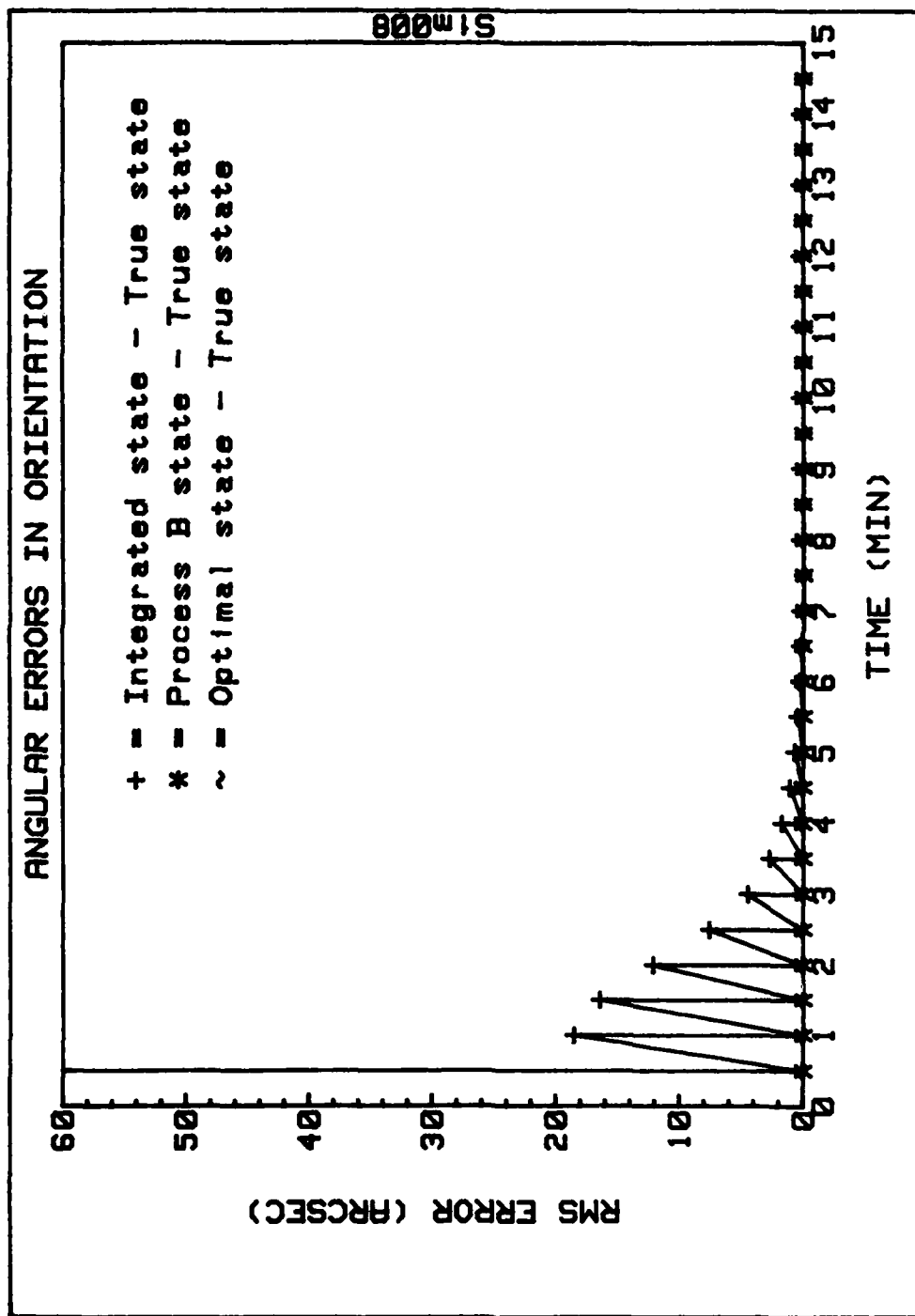


Figure 5.1: Orientation errors of vehicle frame for simulation test with no rate gyro errors and no image centroid errors.

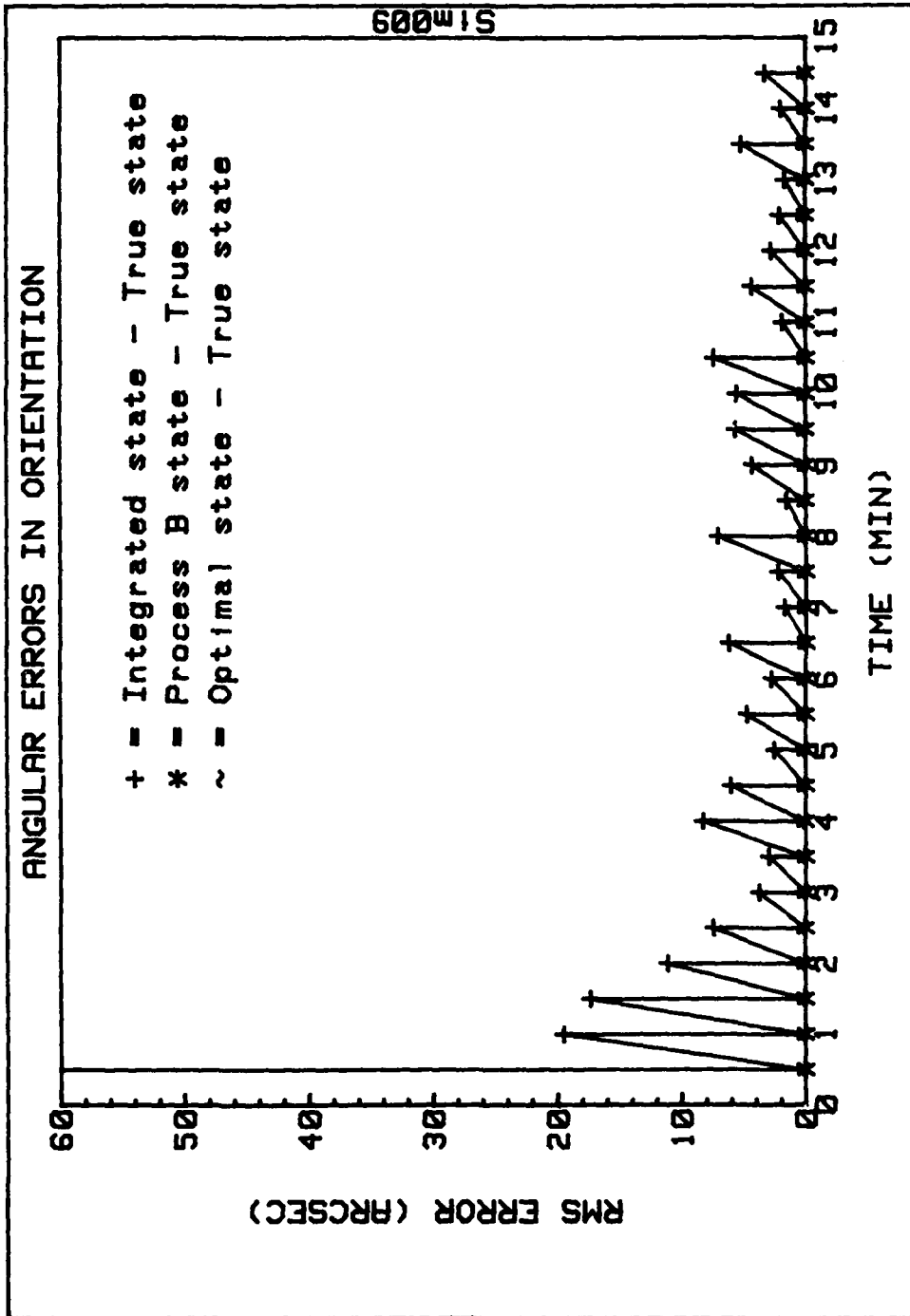


Figure 5.2: Orientation errors of vehicle frame for simulation test with noise added to rate gyro data. ($\sigma = 1$ arc sec/second).

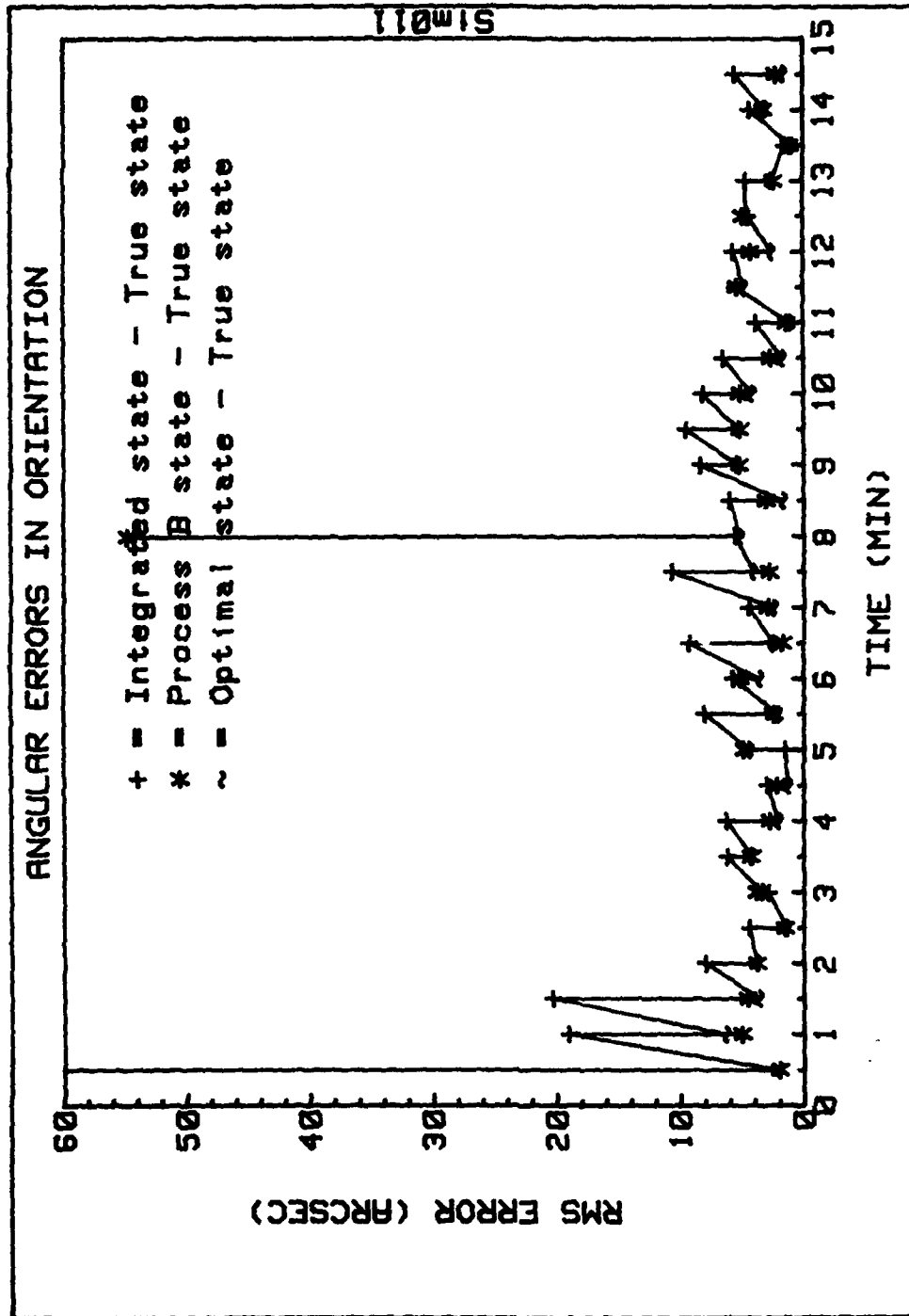


Figure 5.3: Orientation errors of vehicle frame for simulation test with image centroid errors. ($\sigma = 0.0034$ mm).

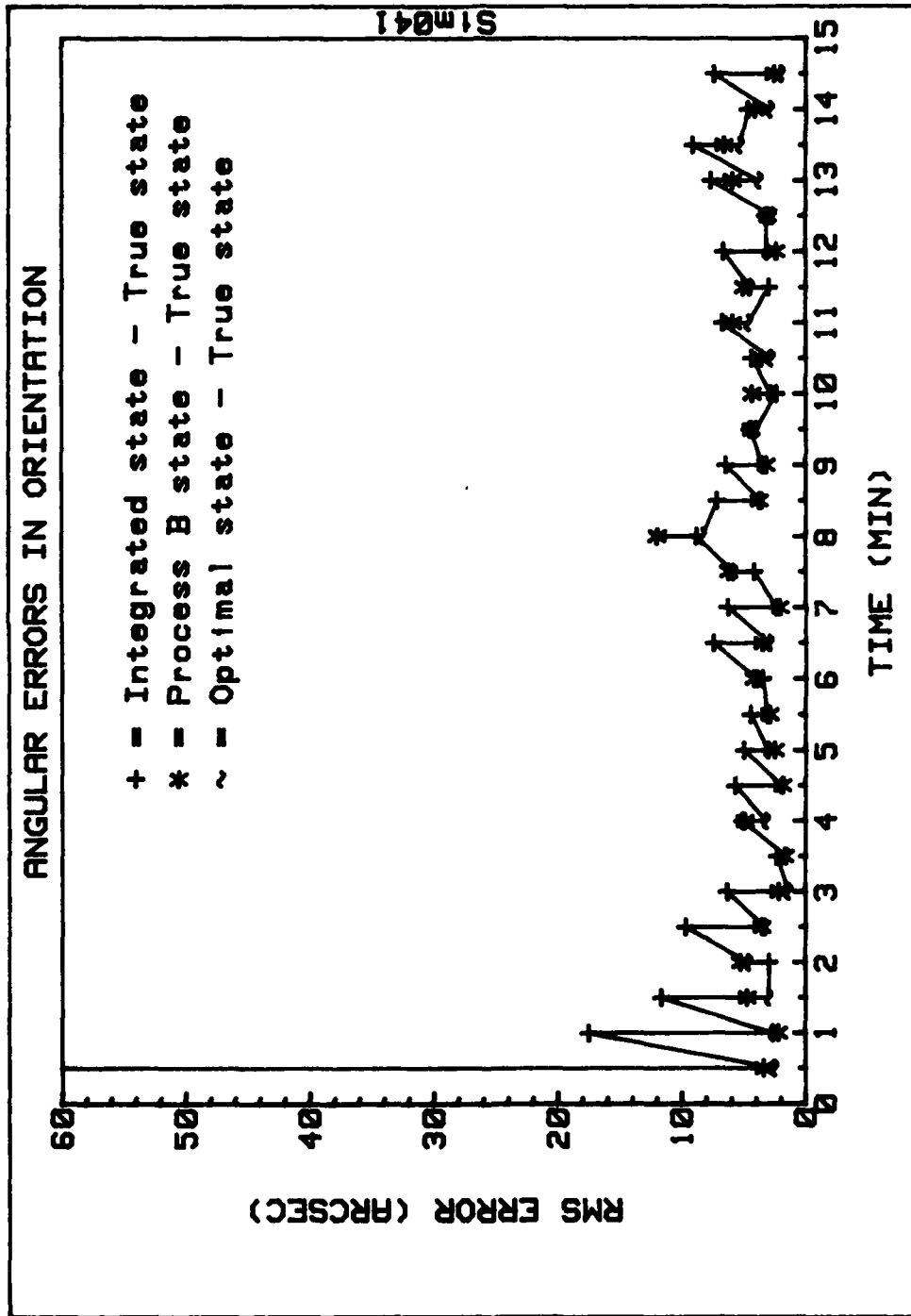


Figure 5.4a: Orientation errors of vehicle frame for simulation test with time varying interlock angles between camera frames. (process noise = 5 arcseconds)

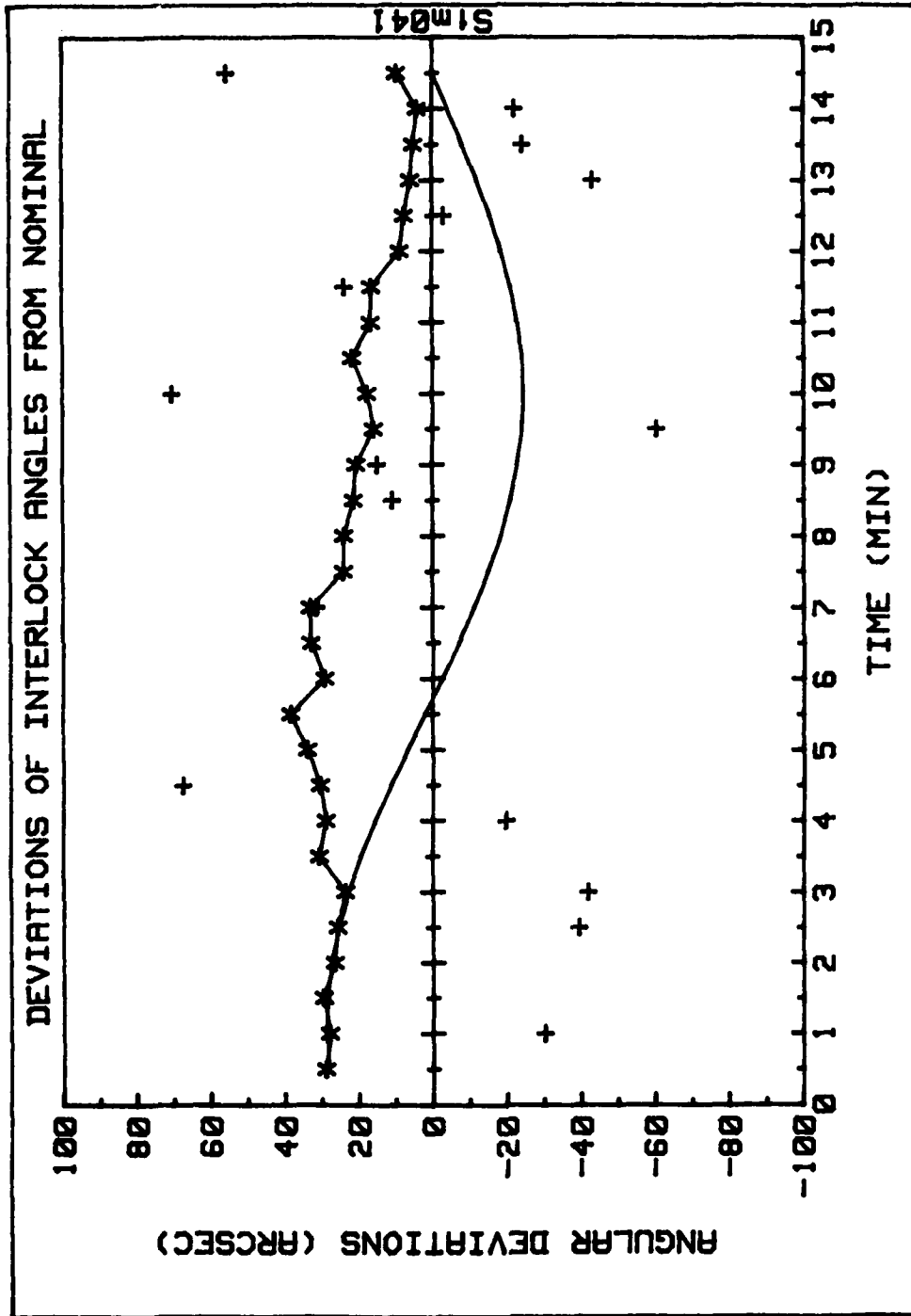


Figure 5.4b: Deviations of the first Euler angle (3-1-3 set) from the nominal value (0°): - = true angle, + = least-squares estimate, * = Kalman filter estimate.

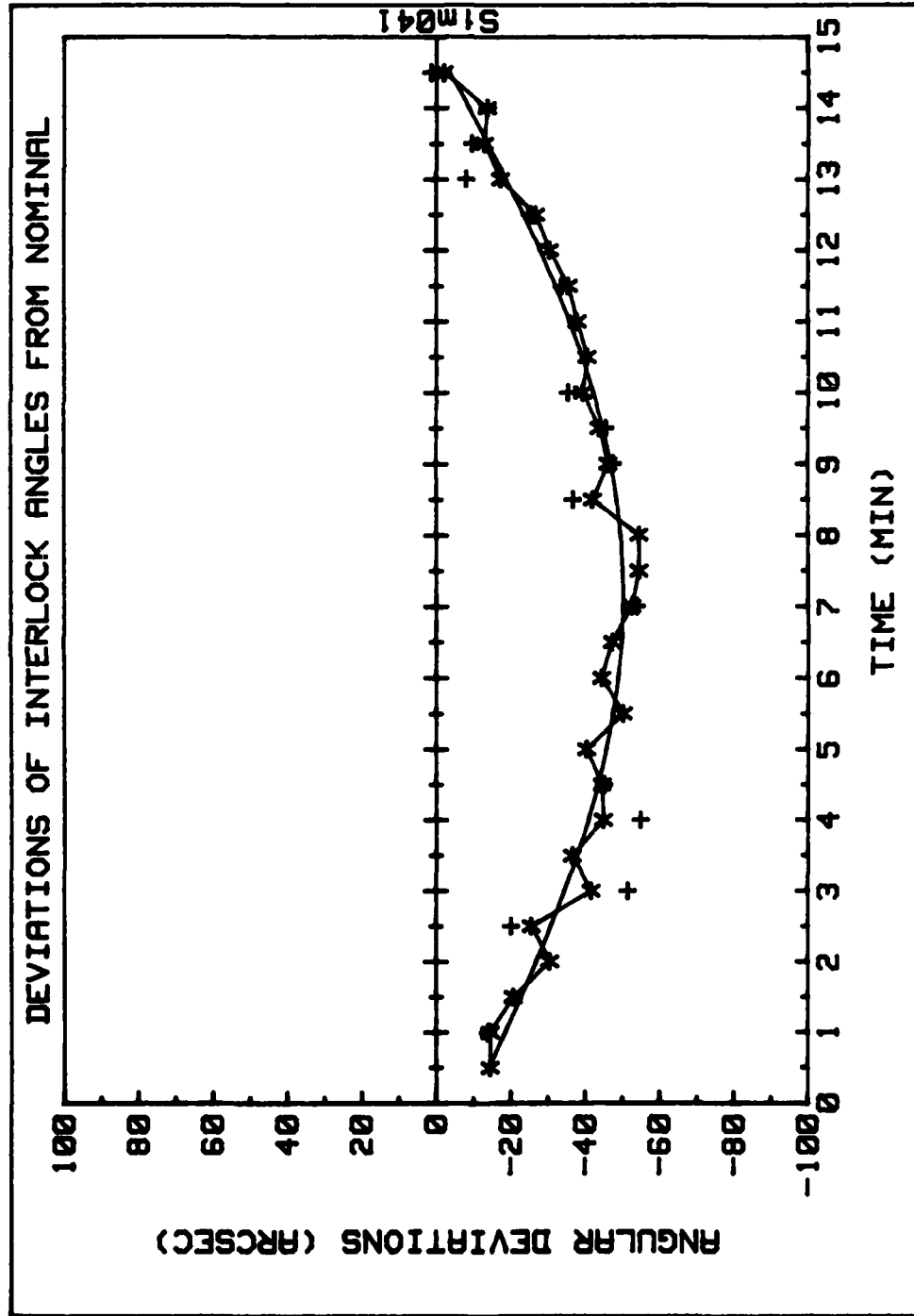


Figure 5.4c: Deviation of the second Euler angle (3-1-3 set) from the nominal value (90°): - = true angle, + = least-squares estimate, * = Kalman filter estimate.

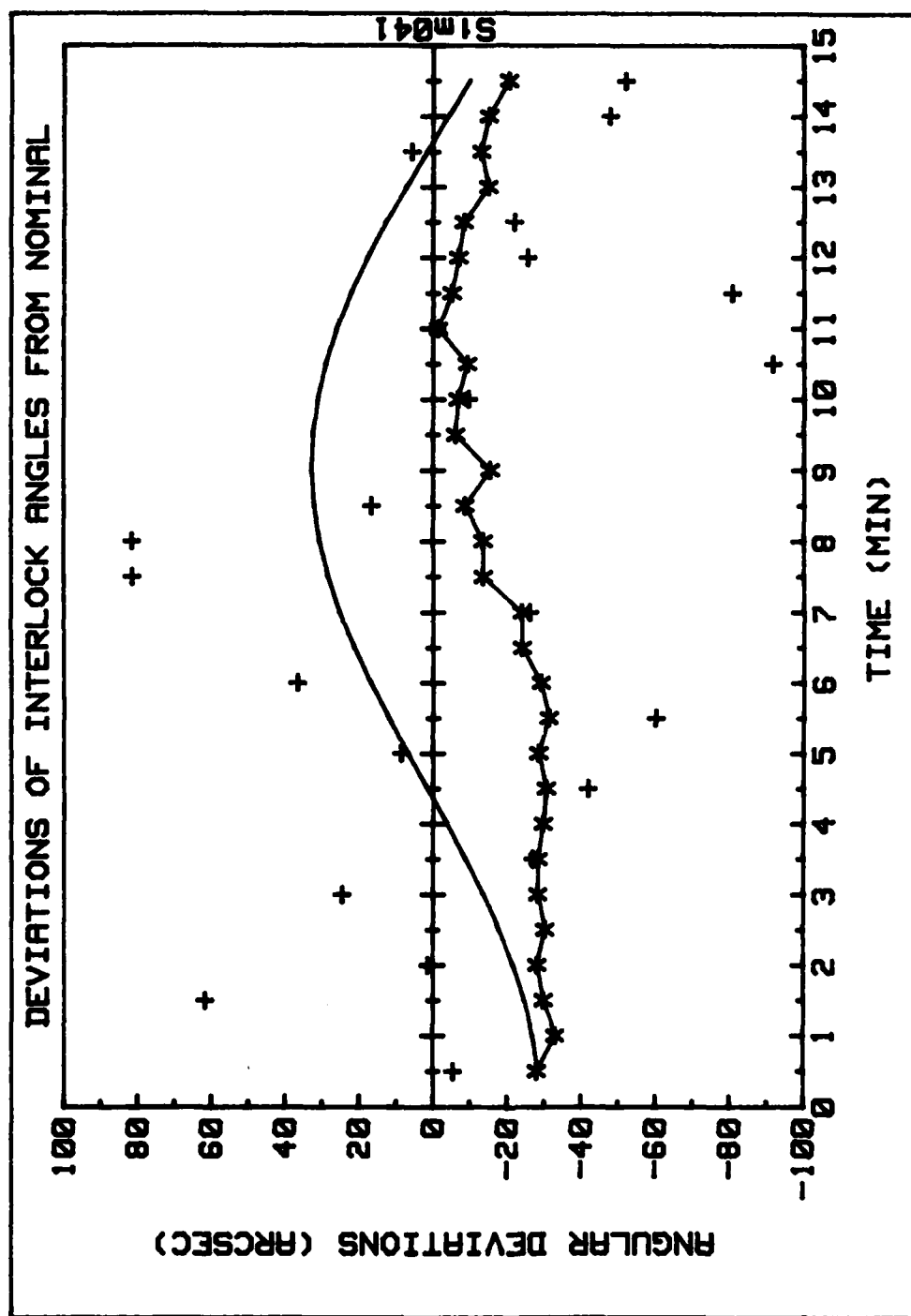


Figure 5.4d: Deviation of the third Euler angle (3-1-3 set) from the nominal value (0°): - = true angle, + = least-squares estimate, * = Kalman filter estimate.

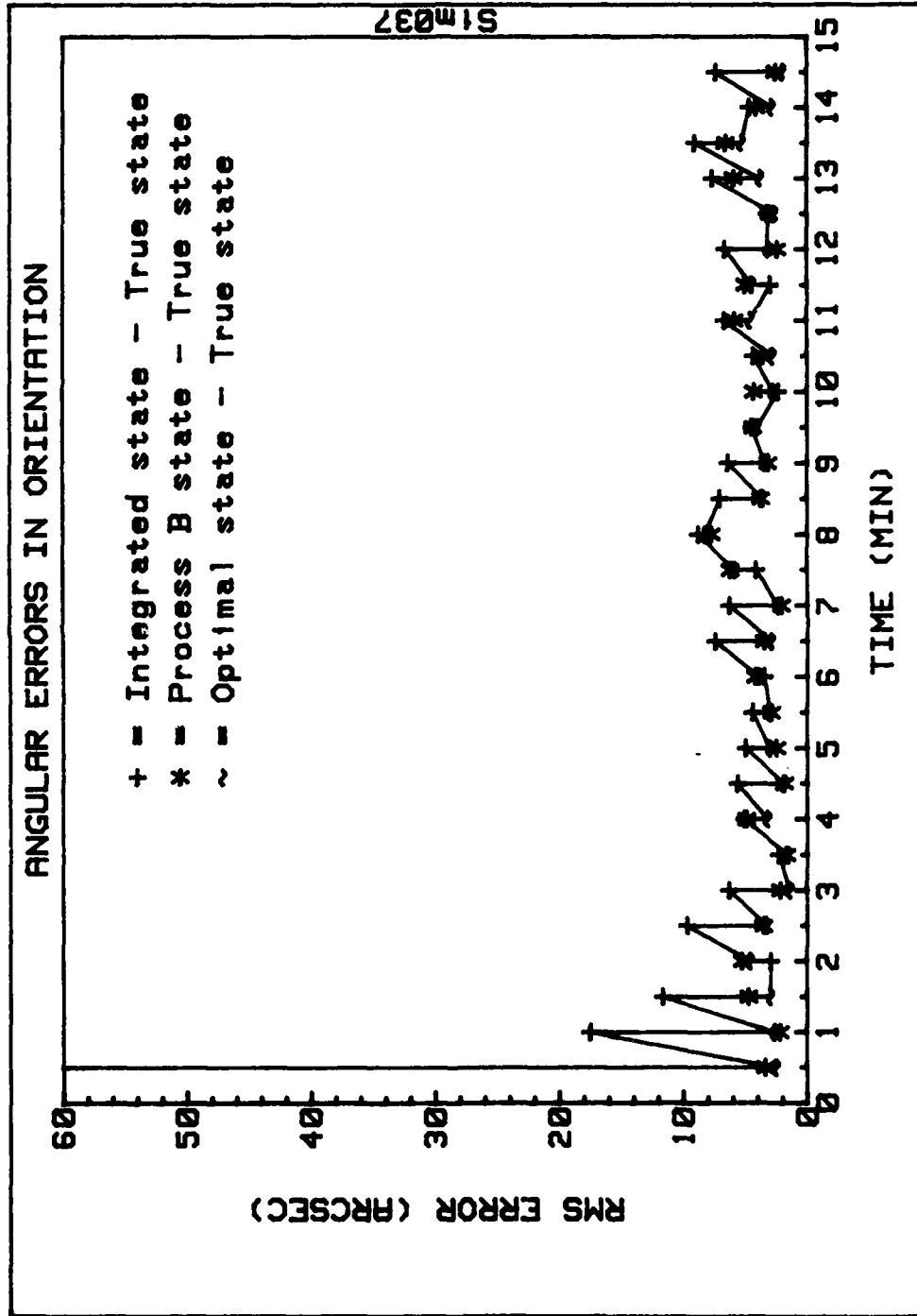


Figure 5.5a: Orientation errors of vehicle frame for simulation test with time varying interlock angles between camera frames. (process noise = 10 arc seconds)

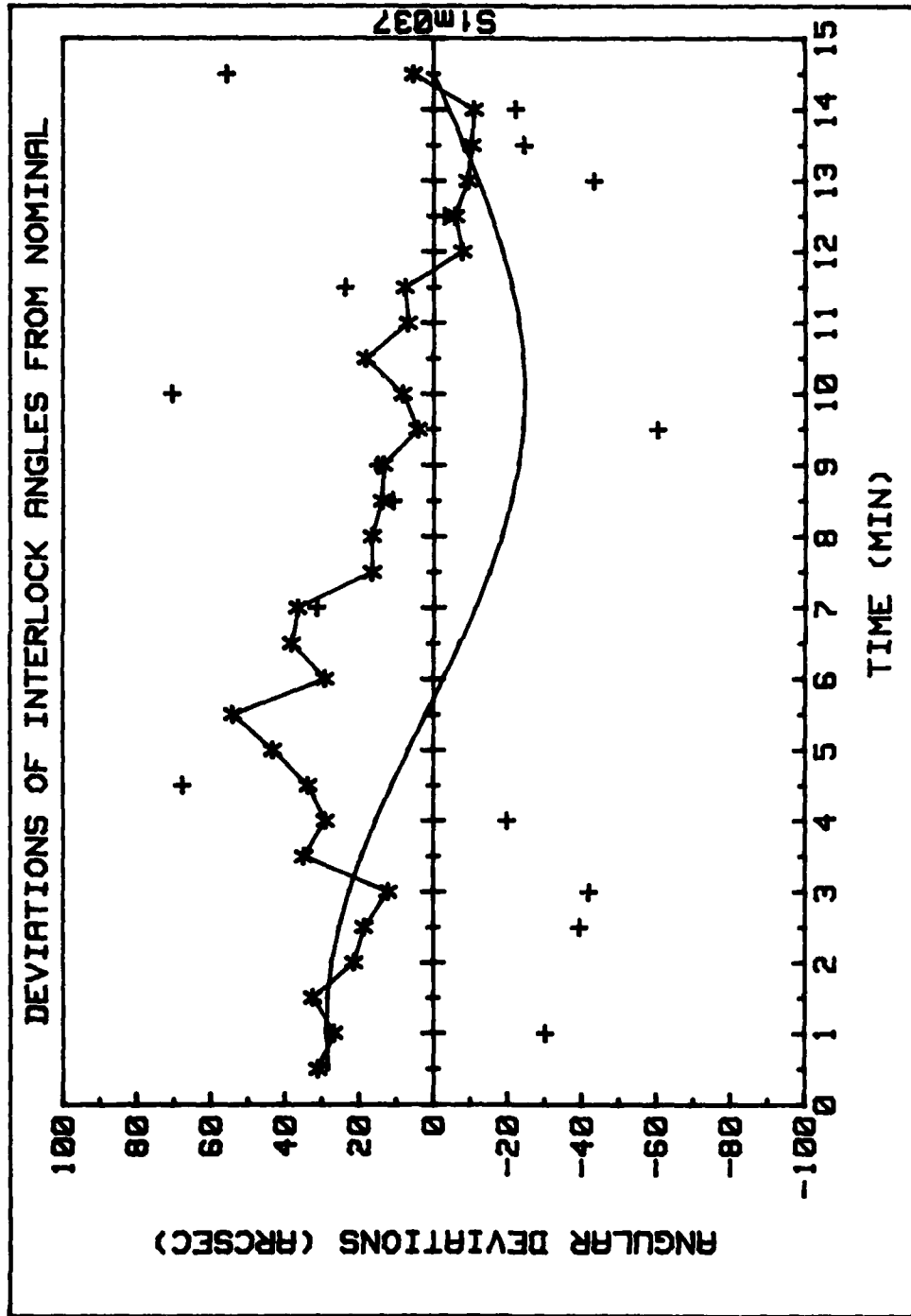


Figure 5.5b: Deviations of the first Euler angle (3-1-3 set) from the nominal value (0°): = true angle, + = least-squares estimate, * = Kalman filter estimate.

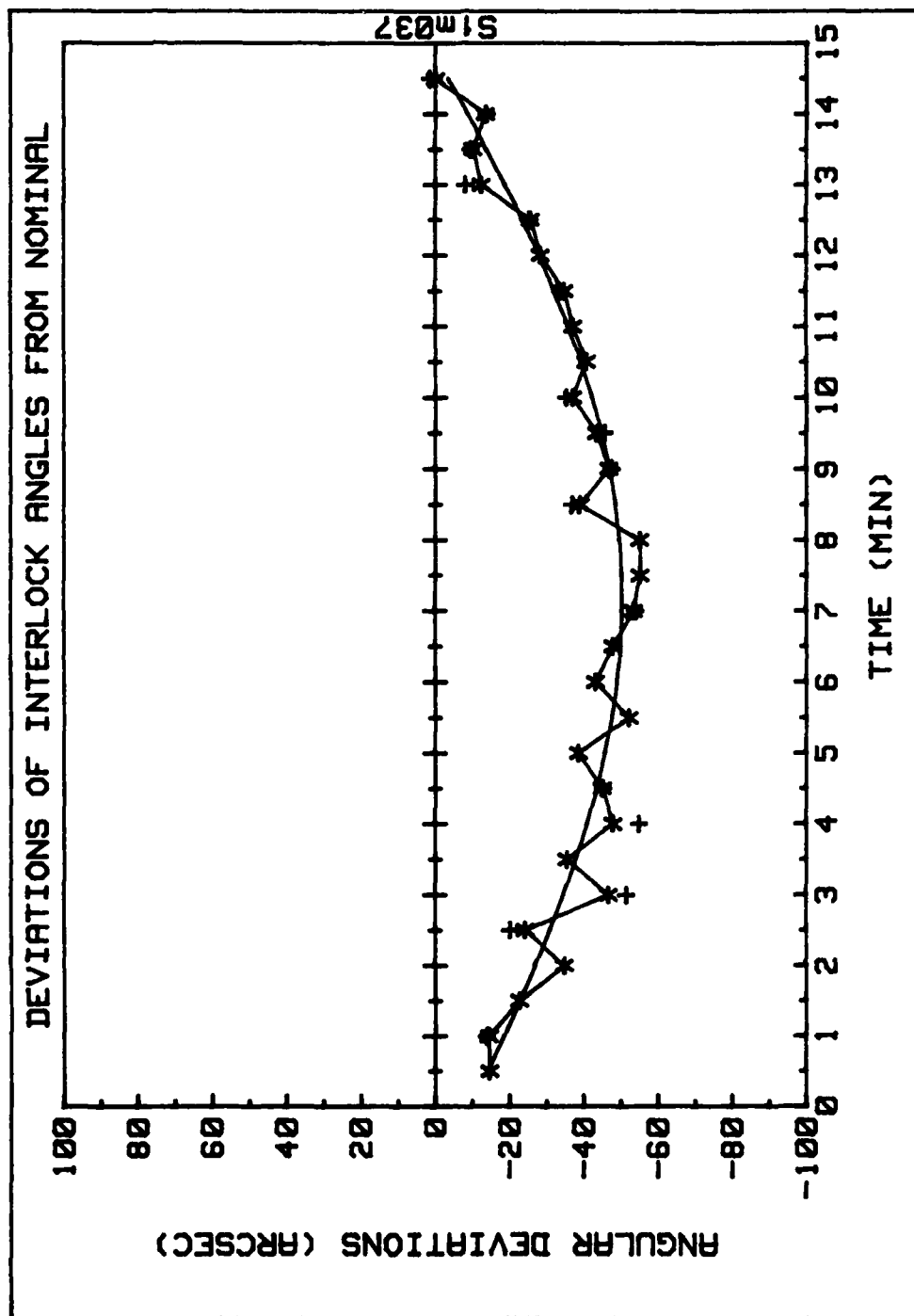


Figure 5.5c: Deviation of the second Euler angle (3-1-3 set) from the nominal value (90°): - = true angle, + = least-squares estimate, * = Kalman filter estimate.

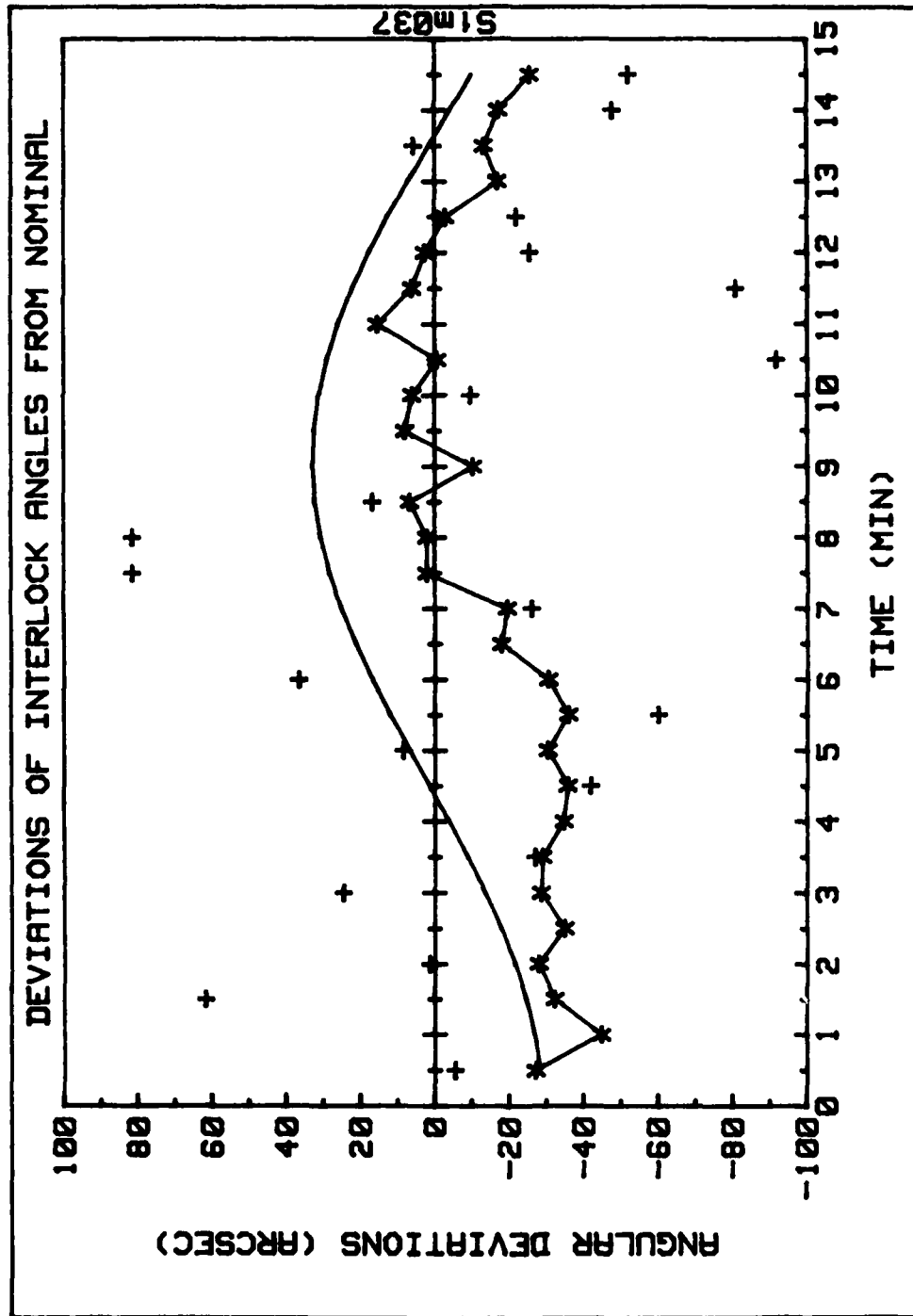


Figure 5.5d: Deviation of the third Euler angle (3-1-3 set) from the nominal value (0°): - = true angle, + = least-squares estimate, * = Kalman filter estimate.

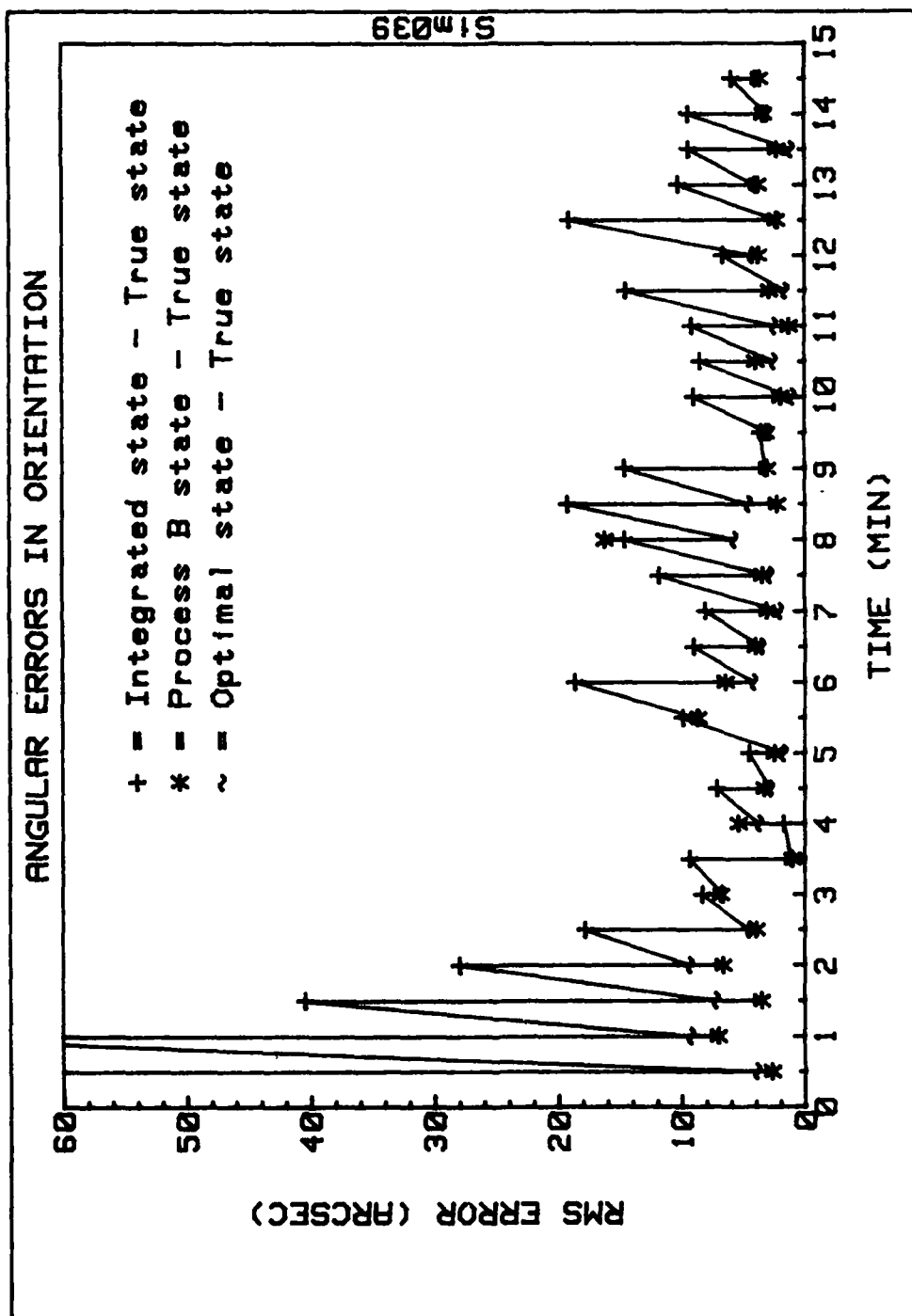


Figure 5.6a: Orientation errors of vehicle frame for simulation test with time varying gyro bias terms.

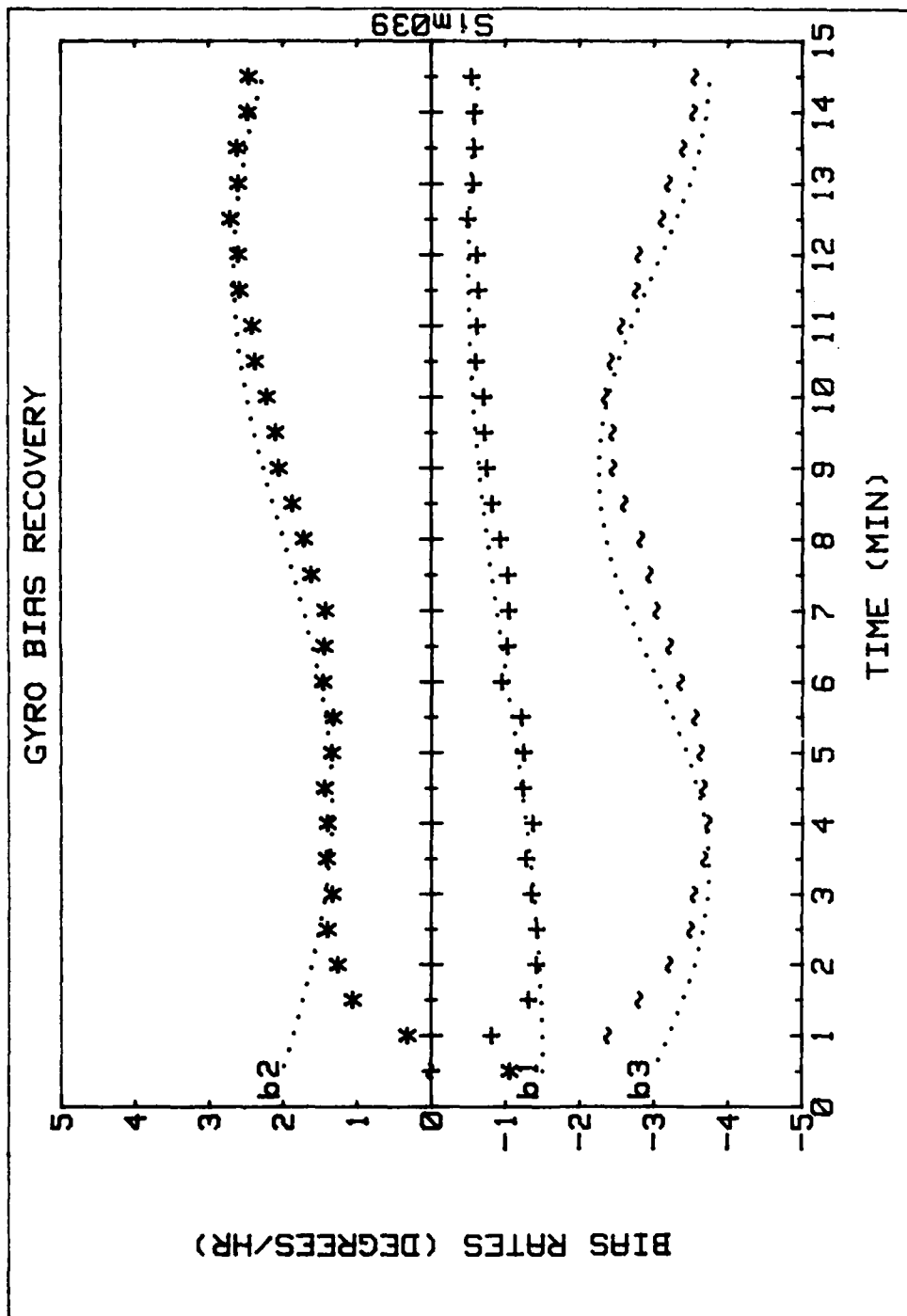


Figure 5.6b: True and calculated bias terms for each gyro axis (bias process noise standard deviation = 0.5 arc sec/second).

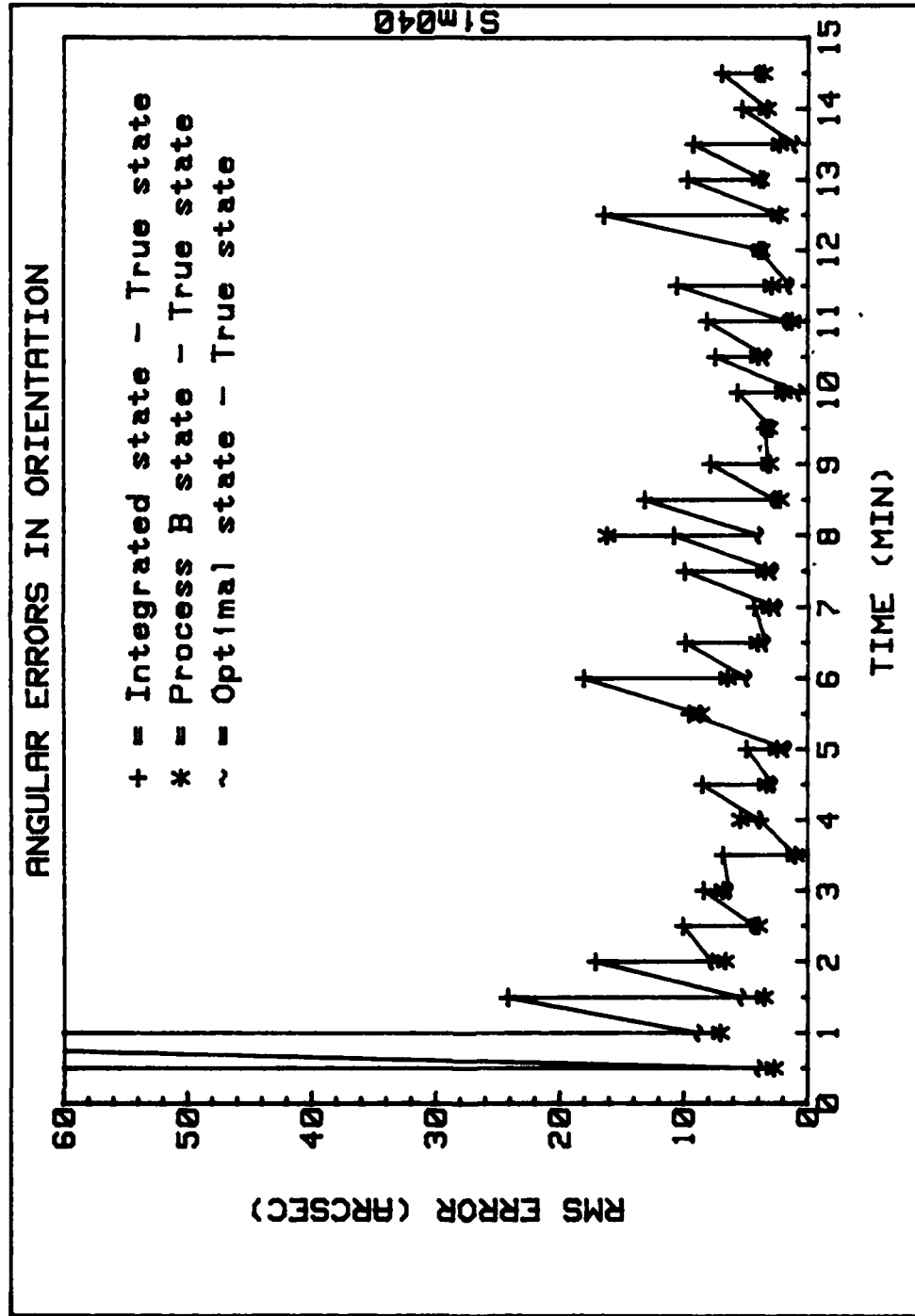


Figure 5.7a: Orientation errors of vehicle frame for simulation test with time varying gyro bias terms.

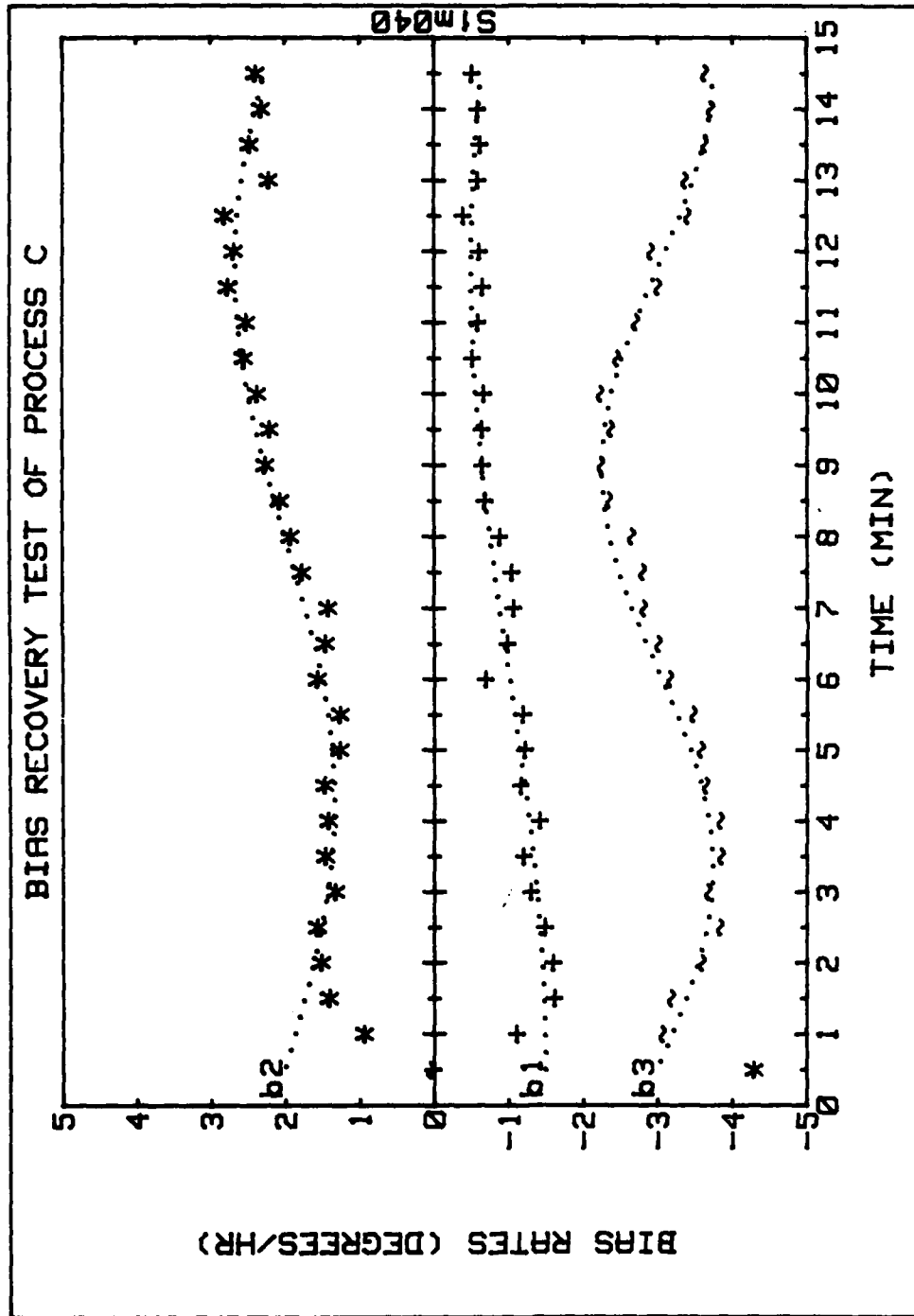


Figure 5.7b: True and calculated bias terms for each gyro axis (bias process noise standard deviation = 1.0 arc sec/second).

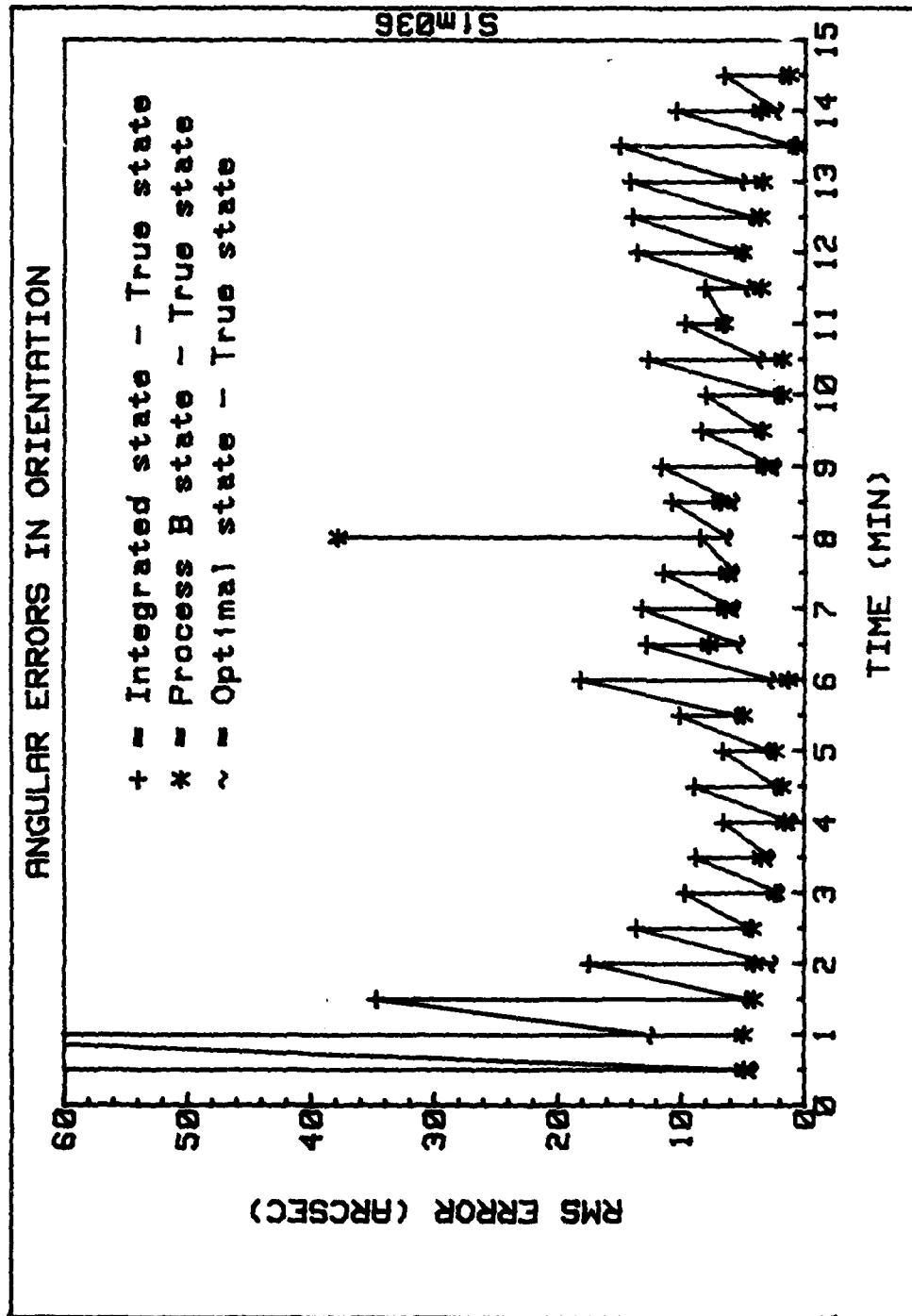


Figure 5.8a: Orientation errors of vehicle frame for simulation test with time varying interlock angles between the gyro and vehicle frame.

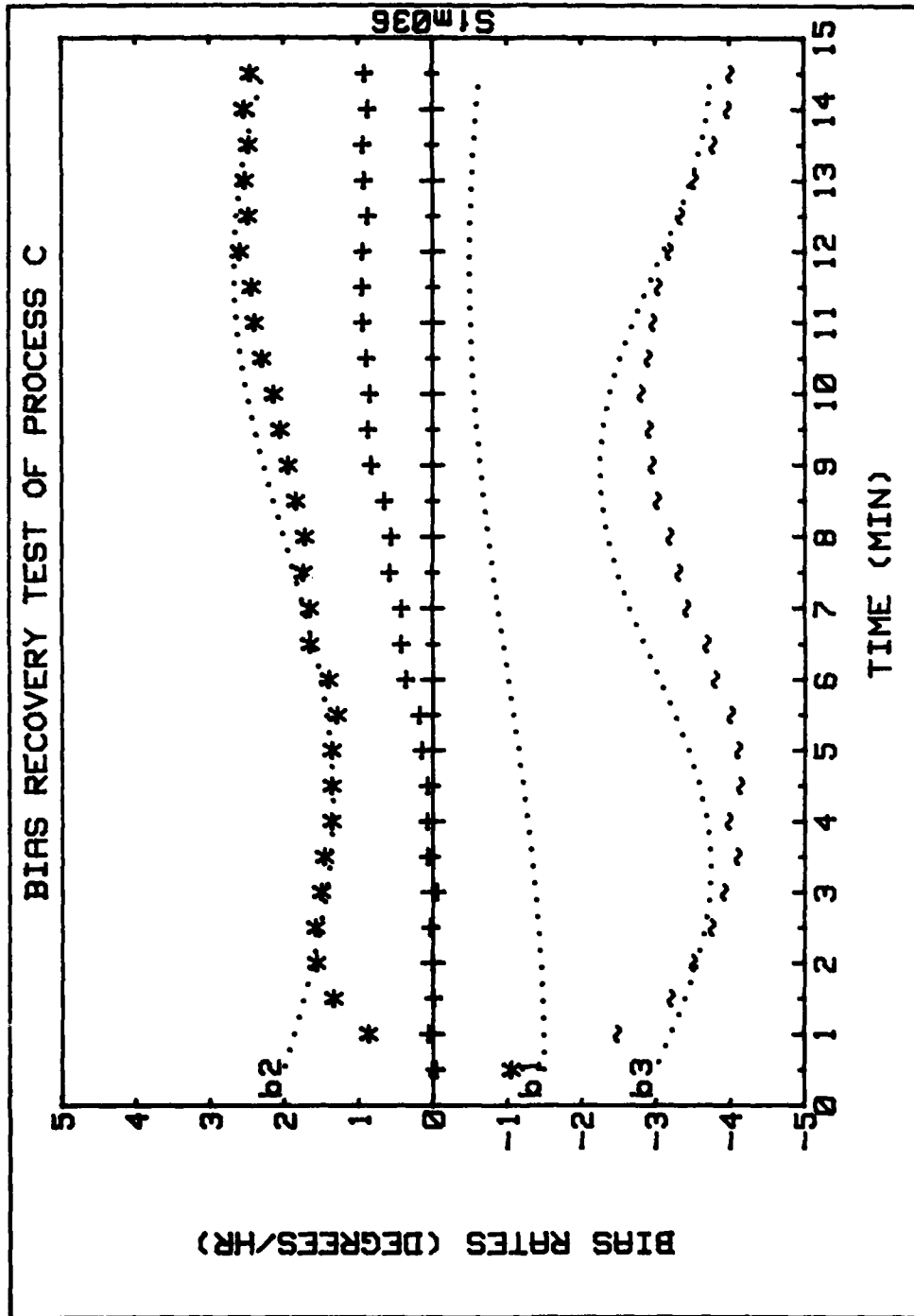


Figure 5.8b: True and calculated bias terms for each gyro axis (bias process noise standard deviation = 0.5 arc sec/second).

6.0 Conclusions

The simulations discussed in Section 5 illustrate that our algorithms can routinely yield 5 arcsecond accuracy for the assumed star tracker configuration and using up to 5 stars in each field of view. There will be occasional errors greater than 5 arcseconds because of too few stars in one or both FOV. Therefore, we consider our quoted accuracy to represent a one sigma error.

We have not demonstrated explicitly that the necessary calculations can be carried out rapidly enough to yield a new attitude estimate every 30 seconds, as planned. However, this can be accomplished, we believe, simply by converting our algorithms from an interpreter to compiler type of computer language. Such a change would probably reduce computer time by a factor of 5 to 10 (from roughly 60 seconds per frame to less than 12 seconds).

There are several features of our algorithms which need special emphasis. First, it is important to keep in mind that our algorithms assume a slowly rotating satellite. Our algorithms are designed to determine the vehicle attitude provided there is an attitude estimate which is within, say, 5-10 degrees of the truth. In a steady-state mode, we can integrate rate-gyro data between successive frames and thereby provide sequential estimates. However, there must be some system such as horizon sensors to provide a rough attitude estimate either in case of start-up, after vehicle maneuvers, and perhaps after successive failures of Process B.

Also, several parameters must be chosen after a real system is designed or assembled. Two of these are the variance for the Kalman

filter interlock estimation and the gyro bias variance for the Kalman filter bias estimation; both of these are important for the self-calibrating features of our algorithms. We have not attempted to provide the extensive error checking capability needed on a flight system since it is impossible to foresee the many types of failures or errors encountered in a real-life situation.

There are, of course, many possible modifications and additions which could be made. One obvious modification is to reduce the number of stars used in the least-squares correction from the current 5 to perhaps 3 or 4 per FOV. Obviously, this will reduce the attitude accuracy but would have the advantage of reducing computing time and memory requirements. Perhaps with some additional logic the 3 or 4 stars most widely distributed over the FOV could be selected and thereby lessen the impact of fewer stars. An improvement in attitude estimation could be obtained by using a longer focal length lens for each star tracker. This reduces the pointing error to each star due to centroiding errors and therefore improves orientation estimates. However, such a change reduces the field of view and the number of stars detected (unless a larger lens is used to detect fainter stars, requiring a larger catalog as well).

7.0 REFERENCES

- 1.0 Junkins, J. L., C. C. White, and J. D. Turner, "Star Pattern Recognition for Real Time Attitude Determination," Journal of the Astronautical Sciences, Vol. XXV, No. 3, Sept. 1977.
- 2.0 Junkins, J. L., Strikwerda, T. E., and Kraige, L. G., Star Pattern Recognition and Spacecraft Attitude Determination, Phase I, VPI&SU Report No. VPI-E-79.4, VPI&SU, Blacksburg, VA 24061.
- 3.0 Strikwerda, T. E., and Junkins, J. L., Star Pattern Recognition and Spacecraft Attitude Determination, Phase II, Interim Report, VPI&SU Blacksburg, VA 24061.
- 4.0 Junkins, J. L., Optimal Estimation of Dynamical Systems, Sijthoff-Nordhoff International Publishers, The Netherlands, ch. 5, 1978.
- 5.0 Gottlieb, D. M. "SKYMAP System Description: Star Catalog Data Base Generation and Utilization," Computer Sciences Corporation, Report CSC/SD-76/6041, Nov. 1976, Silver Springs, MD.
- 6.0 Charged Coupled Devices: Technology and Applications, ed. Roger Melen and Dennis Buss, New York: IEEE Press, 1977.
- 7.0 Johnson, H. L., "Astronomical Measurements in the Infrared," Ann. Rev. Astronomy and Astrophysics, V. 4, 193, (1966).

Appendix 1: CCD Star Tracker

An important element of the attitude determination system discussed in this report is the CCD star tracker. In this appendix we discuss some of the key features of such a star tracker.

A CCD array has a high degree of dimensional stability and is relatively immune to magnetic effects. These features make it very attractive for a star tracker. For a general purpose tracker the field size is typically 5° to 10° wide and with CCD array sizes currently available, the star images would be a fraction of a pixel in diameter. By defocusing the camera lens slightly so a typical image covers a 3×3 array of pixels an image centroid can be computed with at least 10% pixel accuracy. By applying a stored correction function, this error can be reduced still further.

Two other factors affecting centroid accuracy are pixel response variations and photon noise. Response variations can be corrected via a-priori calibration. However, due to time and computer memory limitations only the most severe variations would be corrected in practice. Cooling the CCD reduces thermal noise but the photon noise is always present, affecting fainter images more than brighter images. It is expected that a star tracker with sophisticated software could determine centroids with an accuracy of 5% of a pixel (1σ) for minimum brightness stars and perhaps 2-3% error for the brightest stars.

To prevent the star images from smearing on the CCD due to vehicle motion, the *exposure time* or integration time must be kept short. On the other hand, the analog voltage response from star illuminated pixels must be accurately converted to a digital value, a relatively slow process. Several techniques can be employed to improve read out speed.

The first is to trigger the analog to digital (A/D) conversion only for preselected pixels or those that exceed a minimum threshold. Preselection can be done after a set of stars has been located from the previous frames, while the triggering method can be instituted in a search mode. A second speed gain can be achieved by *line-skipping*--skipping the read-out of rows of pixel responses after they have been transferred from vertical registers into the horizontal register. This technique can be employed in the *track* mode once a set of stars has been located.

We anticipate that Process A will be able to receive multiple frames of data from the star trackers before Process B is ready to accept new data. This may allow Process A time to edit the data such as predicting, crudely, where the stars may appear in the next frame and/or providing Process B with some average position for each star in a frame. This latter technique could improve the projected accuracy by averaging out some random position errors.

Appendix 2: CCD Instrument Response and Stellar Magnitude Conversion

As mentioned in Section 1, the outputs of Process A are the interpolated centroids and instrument magnitudes for each valid star image. The purpose of this Appendix is the discussion of the approximate techniques utilized in the synthesis of these two outputs.

A2.1 The Star Centroids

The centroid location (x_c, y_c) is given by

$$x_c = \frac{\sum_i (x_i \sum_j R_{ij})}{\sum_i \sum_j R_{ij}} \quad (A2.1)$$

and

$$y_c = \frac{\sum_j (y_j \sum_i R_{ij})}{\sum_i \sum_j R_{ij}}, \quad (A2.2)$$

where R_{ij} is the A/D converted response level of the pixel located at (x_i, y_i) and the summations are over the square array of pixels illuminated by the defocused star image (9 to 16 pixels).

Typical cell size for a CCD is approximately 0.030 mm on a side. A CCD placed behind a 70 mm focal length lens (proposed for one CCD star-sensor) gives a resolution of approximately 1.5 arc-minutes for a focused image. When spread over a 3×3 or 4×4 cell pattern, the resolution with which the centroid can be located has been found to be ≤ 6 arc-sec. For double stars, Process A would produce image coordinates for a single star but with poorly determined image coordinates (a weighted mean of the two stars).

Detections of double stars should not be used in Process B since they would result in poor orientations. One solution to this problem

is to delete from the mission catalog all star pairs with separations less than some tolerance (~ 6 arc-min. in this case). There are sufficient stars in the catalog that this deletion should not seriously degrade performance. Some additional time in Process B will be used trying unsuccessfully to pair measured double stars with catalog stars, but since detection of double stars will be a relatively rare event, this time penalty should not be a significant practical problem.

A2.2 CCD Magnitude Response

A2.2.1 Magnitude Conversion

Due to both the different spectral qualities of various stars and the peculiarities in the unfiltered CCD response (the primary sensitivity is to red or near infra-red radiation) two stars of the same visual (V) magnitude (for example) may cause different CCD response. Hence none of the cataloged star magnitudes may be used directly. Rather, the magnitudes of the stars must be properly transformed (using the spectral properties contained in the master star catalog SKYMAP) prior to insertion in the mission catalog. Simply stated, the mission catalog must contain an "instrument magnitude" for each star.

It has been decided to convert the V magnitudes of SKYMAP to I magnitudes and utilize an I filter placed over the CCD array. The following points support this decision:

- (1) Given the information in SKYMAP, one could, in principle, perform a magnitude conversion from V to a CCD magnitude. However, this would require the choice of a particular CCD detector in order to determine its response characteristics in the laboratory.

- (2) The response functions of typical CCD's are quite broad, a fact which makes a rigorous conversion to a CCD magnitude difficult in light of the complicated stellar spectral features in the blue wavelength region. A detailed description of the spectra would be required. The I filter, on the other hand, is confined to the red wavelengths where the star spectra are relatively smooth.
- (3) The I filter response peak is near that of typical CCD's. In addition, it overlaps the main peaks of the commonly accepted "typical CCD response" (Ref. 6). Hence, an I filter placed over the CCD array would serve to limit the wings of the CCD response and still provide adequate through-put for sensitivity.
- (4) Information exists for converting V magnitudes to I magnitudes. The transformation requires only spectral type and luminosity class - both readily available from SKYMAP.

In the ideal case, the set of detectable stars exactly matches the catalog. Since this is not possible, it is desirable to maximize the completeness of the catalog to some rather faint magnitude to insure that most detectable stars are contained. It is important to note that stars which are faint in V may be relatively brighter at red wavelengths. As will be demonstrated in the next sections, a limit of magnitude 5 in I seems to be a reasonable limit for the CCD configuration assumed for the present study. The 8th magnitude limit of SKYMAP assures that the mission catalog listing will be sufficient.

A2.2.2 The I Magnitude Conversion Method

Two external items of information are needed for the magnitude conversion. Values of $V_{\text{abs}} - I_{\text{abs}}$ were obtained from Johnson [7], whose

table contains listings for three luminosity classes (I, III & V - super giants, giants and main sequence) and extends over most spectral types. The second value needed is a_I/a_V , the ratios of absorption in I to absorption in V, expressed in magnitude and as a function of spectral type by

$$\frac{a_I}{a_V} = \frac{\log_{10}[(\int_0^\infty I(\lambda)E(\lambda)(1 - I(\lambda)d\lambda)/(\int_0^\infty I(\lambda)E(\lambda)d\lambda)]}{\log_{10}[(\int_0^\infty V(\lambda)E(\lambda)(1 - I(\lambda)d\lambda)/(\int_0^\infty V(\lambda)E(\lambda)d\lambda)]} \quad (A2.3)$$

Where $I(\lambda)$ and $V(\lambda)$ are the filter responses, $E(\lambda)$ is the star energy function and $I(\lambda)$ is the relative absorption function and λ is the wavelength. (See SKYMAP description [5] for details). As pointed out in SKYMAP, this ratio is nearly constant over spectral type for narrow or intermediate band filters. To calculate this ratio the Planck energy function was used to model the stellar flux. Although this is not precisely valid, forming the ratio should lead to quite accurate results. The temperatures used were those given by Johnson [7] and no distinction was made by luminosity class. The values for absorption were taken from Fig. 3.2 in the SKYMAP description by assuming absorption in magnitude is a linear function of wavelength over the range of interest (4800 Å - 10000 Å),

$$a(\lambda) = 1.77 \times 10^{-4} \times \lambda(\text{Å}) + 1.77, \quad (A2.4)$$

Where $a(\lambda)$ is absorption in magnitude at wavelength λ . The results of these calculations were that a_I/a_V varied from 0.25 to 0.32. The same value of a_I/a_V was used for I, III and V luminosity class stars at a given spectral type.

A2.2.3 SKYMAP DATA

The data from SKYMAP needed for the magnitude conversion consists of apparant visual magnitude, spectral type, luminosity class and

absorption in V. These data, with the exception of a_V , are given for most stars. It was necessary to collapse categories of luminosity class and some spectral types since the table from Johnson is limited. Collapsing is justified in most cases because either the category contains few stars and/or the properties are similar to those of listed star types. The following combinations were used:

R	}	Luminosity class III, spectral type M (all are variants of M III)
N		
C		
S		
WR	}	Lum class III, spectral type 09 (all are hot giant stars similar to 09)
WC		
WN		
IV	}	III
III		
II		
V	}	V
VI		
I	}	I
I _{a,ab,b}		

No Luminosity class \rightarrow V (most stars are V stars)

No subinterval in spectral type \rightarrow 5 (i.e., A becomes A5)

No absorption given \rightarrow set to 0

No spectral type \rightarrow exclude

Given the spectral type and luminosity class, the value of $V_{\text{abs}} - I_{\text{abs}}$ and a_I/a_V were found by interpolation in the table. Then:

$$m_I = m_V - (V_{abs} - I_{abs}) - a_V(1 - a_I/a_V), \quad (A2.5)$$

where m_V and a_V are from SKYMAP.

SKYMAP contains approximately 45,000 stars. Of these, 37 were not processed due to missing spectral type or visual magnitude.

A2.2.4 Magnitude Limit of CCD Sensor

In order to establish a reasonable magnitude limit for a CCD sensor we 1) determine the flux in the filter bandpass for some standard star at the earth's atmosphere, and 2) multiply by appropriate factors dictated by the sensor.

The most direct way to obtain a flux estimate would be to observe known stars from space with the CCD sensor. Barring this, ground-based observations of stars with varying zenith angles could yield flux estimates outside the earth's atmosphere.

Our approximate method was to numerically integrate the surface flux distribution of a K7 V star model atmosphere over the I bandpass. The absolute I magnitude of such a star is approximately 6.2 ($V = 8.1$, $V - I = 1.92$). The radius is given by $\log R^*/R_{sun} = -0.11$ where $R_{sun} = 6.96 \times 10^{10}$ cm. The surface flux is scaled by $(\frac{R^*}{R_{sun}} \times \frac{R_{sun}}{10_{pc}})^2$ $= 3.08 \times 10^{-18}$ where $1_{pc} = 3.08 \times 10^{17}$ cm. The result of integrating and scaling is:

~ 1024 photons/cm²sec.

Typical scale factors are:

-Lens area: 26.7 cm²

-CCD response peak efficiency = .60

-I filter transmission peak efficiency = 0.85

-CCD effective area utilization = 0.46

-Integration time = 0.1 sec.

If we desire a minimum of 7500 photons/star for a sufficient signal-to-noise ratio, we compute the I magnitude limit of:

$$m_{\text{limit}} = 6.2 + 2.5 \log \frac{641}{7500} = 3.5$$

Note that many factors are uncertain or could be altered. Integration time could be increased to 1 sec to give a limit of 6.0. We have chosen 5.0 to be the cutoff magnitude since this seems obtainable and gives approximately 5400 stars, a sufficient number for the pattern recognition process to work reliably.

We also note that model atmospheres for a variety of spectral types could be used to repeat the above calculation to yield a more precise magnitude limit.

The magnitude limit is flexible since the integration time for the star sensor is variable over a wide range. If the integration time is changed by a factor of 10 the magnitude limit is changed by 2.5 mag. In addition, the dynamic range to typical CCD arrays is 200 or about 6 magnitudes. The response is linear over the range to allow accurate magnitude calibration and detection.

Appendix 3: Star Data Base And Mission Catalog Creation

The star catalog data base system SKYMAP (Ref. 5) has been selected as the master star data base. The SKYMAP catalog was developed from the SAO catalog and other sources specifically for attitude determination programs by NASA-GSFC. It is complete to the eighth magnitude in either the blue (B) or visual (V) magnitudes. Additionally, the catalog contains right ascensions, declinations, and, when known, the spectral type, luminosity class, and amount of interstellar absorption in the V wavelength range. Recent work [at the Naval Surface Weapons Center] has uncovered a significant number of corrections to the SKYMAP data base which will be reflected in future revisions of the present SKYMAP data base.

The on-board (or mission) star catalog is divided into celestial sphere cells so as to permit efficient microcomputer access during the pattern recognition process. In order to keep storage requirements for the mission catalog to a minimum, the cells do not overlap. The placement of the cell centers is given by the polar angle θ and longitude λ according to

$$\theta_n = \cos^{-1}(\xi_n) \quad n = 0, 1, 2, \dots, N \quad (A3.1)$$

and

$$\lambda_{nj} = \frac{2\pi j}{2n+1} \quad j = 0, 1, 2, \dots, 2n \quad (A3.2)$$

$$\xi_n = (-1)^n \cos\left(\frac{n\pi}{2N+1}\right), \quad n = 0, 1, 2, \dots, N \quad (A3.3)$$

These formulae yield $(N+1)^2$ points: $N+1$ polar angles or declination zones with spacing $2\pi/(2N+1)$, and $(2n+1)$ equally spaced regions in each zone.

The choice of N is somewhat arbitrary. A large N yields small cells which would require more than one cell to be accessed; a small N yields large cells which would increase the number of trials in the pattern recognition process as well as causing a possible storage problem. Taking into account the $7^\circ \times 9^\circ$ field-of-view, a value of $N = 22$ was chosen, yielding 529 cells.

To facilitate computer access, the cells are ordered within memory according to a parameter n^2+j ; a table lists the starting relative address of each cell and the number of stars in each. Thus, given a boresight estimate (θ, λ) , the primary cell location is given by

$$\begin{aligned} n &= 2[\theta/\Delta\theta + 0.5] \quad (\theta < 90^\circ) \\ &= 2N + 1 - 2[\theta/\Delta\theta + 0.5] \quad (\theta > 90^\circ) \end{aligned} \quad (A3.4)$$

$$j = [\lambda/\Delta\lambda + 0.5], \quad (A3.5)$$

where $[x]$ indicates integer arithmetic (truncation to next smallest integer). The table of cells is then consulted for identification of the appropriate memory location. In all, the catalog access routine reads data from the 4 nearest neighboring cells around the estimated boresight (Figure A3.1) and thus provides nearly complete coverage of the estimated FOV by the 4 cells.

The CCD is assumed (see Appendix 2) to respond to stars of I magnitude 5 or lower - approximately 5400 stars. If these 5400 stars are assumed to be distributed uniformly over the celestial sphere, the star density ρ would be

$$\begin{aligned} \rho &= \frac{5400 \text{ stars/sphere}}{41,253 \text{ square degrees/sphere}} \quad (A3.6) \\ &\approx 0.13 \text{ stars/square degree} \end{aligned}$$

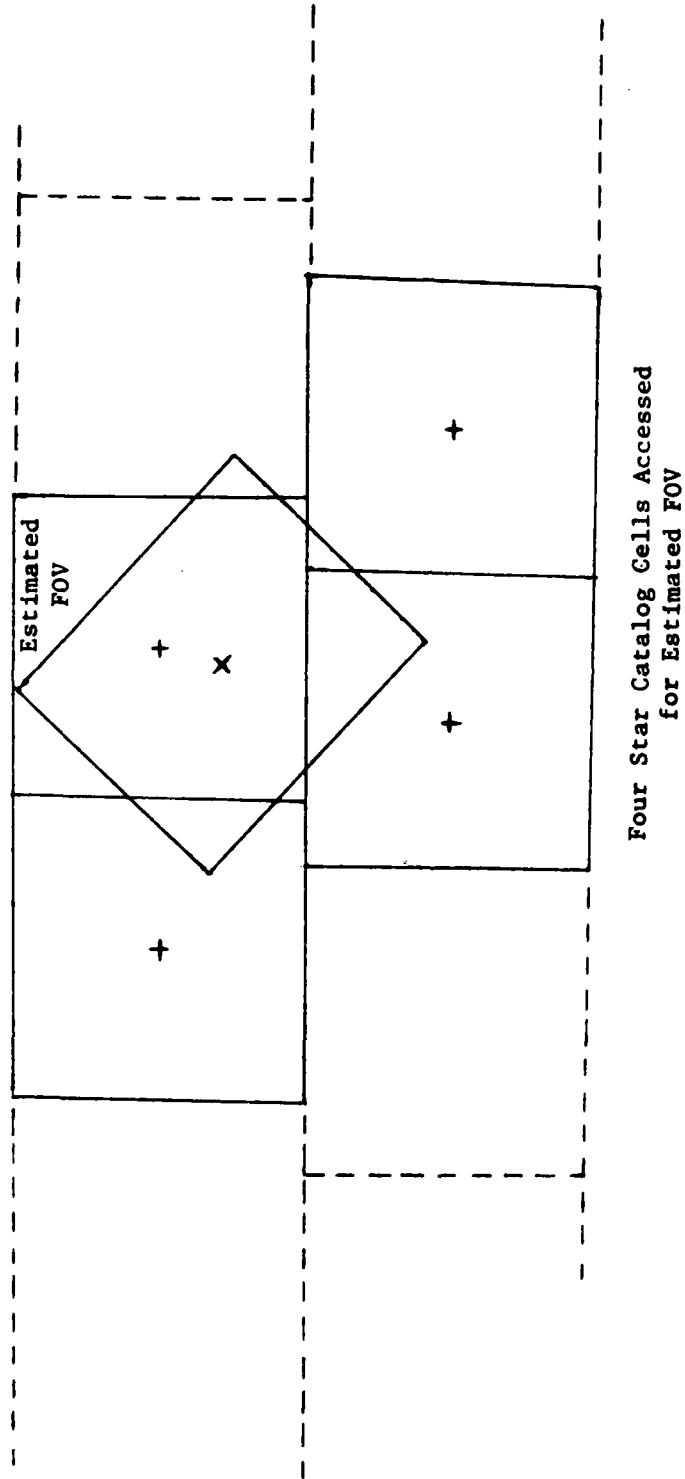


Figure A3.1 Catalog cell pattern obtained for a typical estimated field of view.

For a field-of-view of $7^\circ \times 9^\circ \cong 63$ square degrees, we would expect

$$(63 \text{ square degrees})(0.13 \text{ stars/square degree}) \cong 8.2 \text{ stars}^*$$

in a field-of-view (assuming uniform density). To obtain a measure of the range of the number of stars actually detectable per field-of-view, the boresight was randomly oriented over the entire celestial sphere 100 times. For each trial, the mission catalog was consulted and the number of stars in the field-of-view recorded. The average number of stars per field-of-view was six; in no case were fewer than two stars in the field-of-view.

*Due to non-uniform star population of the celestial sphere, this number decreases to about 5 at the north galactic pole.

Appendix 4: Inter-Star Cosine Calculations

The key to efficient star identification is to take advantage of the sub-ten-arcsecond precession of Process A; the angles between pairs of measured stars are very well determined by the measured coordinates and can be used to identify the corresponding catalog stars. The cosine of the angle between a typical pair of measured stars can be computed from the measured image coordinates as

$$c_{ij} \equiv \cos \theta_{ij} = \frac{x_i x_j + y_i y_j + f^2}{\sqrt{(x_i^2 + y_i^2 + f^2)(x_j^2 + y_j^2 + f^2)}} \quad (A4.1)$$

The cosine of the angle between a typical pair of catalog stars can be computed from the catalog direction cosines as

$$C_{IJ} \equiv \cos \theta_{IJ} = L_{I1} L_{J1} + L_{I2} L_{J2} + L_{I3} L_{J3}. \quad (A4.2)$$

The pattern recognition logic we developed makes use of the smallness of the difference between (A4.1) and (A4.2) as a means to tentatively identify measured stars in the catalog. Our strategy assumes a steady-state condition in which the estimated boresight is within a degree or so of the true boresight direction. Thus, the highest probability of finding a pair match lies in comparing stars from the center of the sub-catalog distribution. For this reason, we sort the sub-catalog stars by angular distance from the boresight. We proceed to pair catalog stars by using the sum of the star indices (after sorting) as our criterion for the pairing order. Each catalog pair is compared with the pairs of measured stars (cosines are stored in a table). We eliminate from consideration star pairs with separation less than one degree because of the possible large "roll" error about

the boresight. In addition, we do not use catalog pairs with separations greater than about 10 degrees (greater than the FOV size). If we find agreement between a catalog pair and measured pair we perform a magnitude test to resolve the 180° ambiguity.

The above strategy is not necessarily optimal. However, we have found it to be very efficient and it allows for a mismatch of several degrees, at least, between the estimated and true boresight vectors.

Appendix 5: Stellar Aberration

The effect of stellar aberration is to cause a star's apparent direction to shift towards the direction of the observers motion. The amount of shift depends on both the velocity of the observer and on the angle between the observer's line of sight (the star direction) and the velocity vector. The shift is:

$$a = \frac{v}{c} \sin \alpha \quad (\text{A5.1})$$

where a = aberration in radians

v = observer's speed

c = velocity of light

α = angle between velocity vector and the true star direction.

For our purpose, we must express a star's shifted direction in terms of the true direction, the vehicle velocity and the angle between the velocity vector and true direction; that is,

$$\begin{Bmatrix} L'_x \\ L'_y \\ L'_z \end{Bmatrix} = f(L_x, L_y, L_z, v, \alpha) \quad (\text{A5.2})$$

If we let \underline{v}_s be the velocity of starlight in the inertial frame and \underline{v} be observer velocity, then the relative velocity of the starlight as seen by the observer is:

$$\underline{v}_{s/O} = \underline{v}_s - \underline{v} \quad (\text{A5.3})$$

Now, if we let $\hat{\underline{l}}_n$ be the unit vector in the true direction and $\hat{\underline{l}}'_n$ the unit vector in the shifted direction, we can rewrite this as (refer to

Figure A5.1)

$$(c + v \cos \alpha) \hat{\underline{\ell}}'_n = c \hat{\underline{\ell}}_n + \underline{v} \quad (\text{A5.4})$$

To first order this becomes

$$\hat{\underline{\ell}}'_n = (1 - \frac{v}{c} \cos \alpha) \hat{\underline{\ell}}_n + \frac{\underline{v}}{c} \quad (\text{A5.5})$$

or

$$\begin{Bmatrix} L'_x \\ L'_y \\ L'_z \end{Bmatrix} = (1 - \frac{v}{c} \cos \alpha) \begin{Bmatrix} L_x \\ L_y \\ L_z \end{Bmatrix} + \frac{1}{c} \begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix} \quad (\text{A5.6})$$

This equation is used for calculating the displacement of a star's unit vector. The velocity vector is computed for the combined velocity of the earth and satellite and Herrick's "f and g" solution is used to calculate the individual velocities each time Process B accepts new data from Process A.

We note several points concerning the effects of aberration on the star tracker. The speed of the earth in its orbit is 30 km/s and the maximum speed of an earth orbiting satellite is < 8 km/s relative to the earth. Therefore, the maximum shift in a star's direction is about 26 arcseconds. This maximum occurs for stars 90° from the velocity vector. However, all stars in this neighborhood will be shifted by nearly this amount and, thus, the distortion of the FOV will be insignificant. However, aberration will displace the boresight direction. To avoid orientation errors in the combined FOV(A) and FOV(B) solution we must correct the catalogue direction cosines by applying Eq. (A5.6).

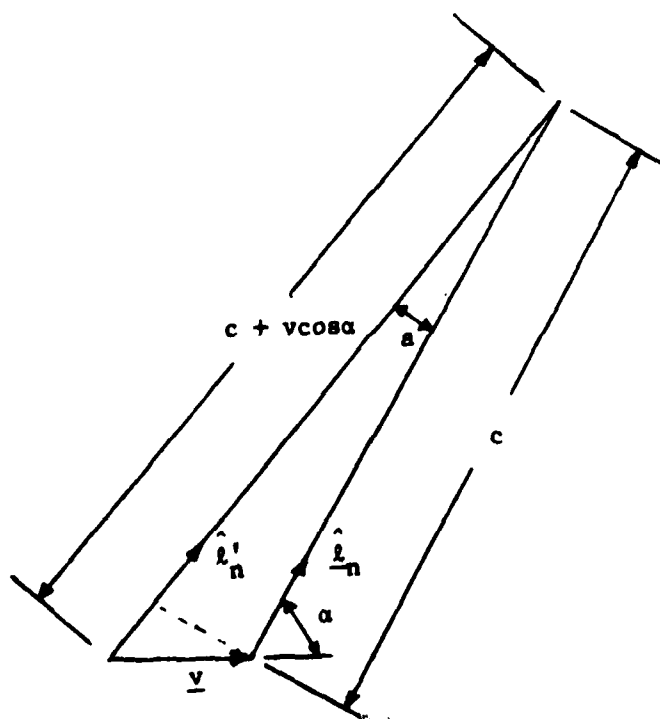
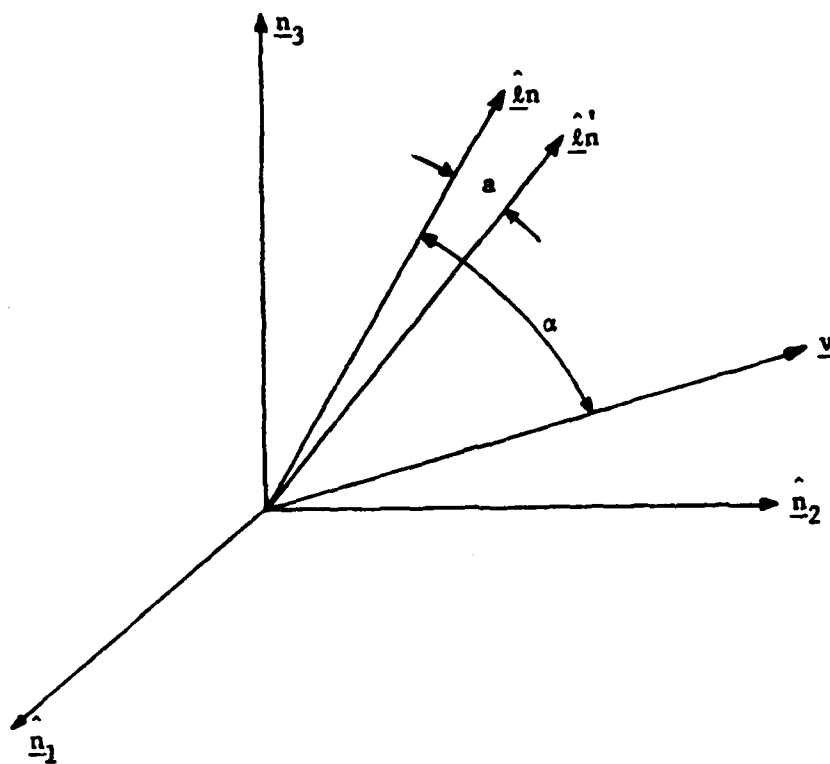


Figure A5.1 Star direction displacement of stellar aberration due to observer's velocity.

For those stars in the direction of the velocity vector, the shift in direction will be small. But since the shift is always towards the velocity vector the distortion is noticeable (an apparant shrinking of the FOV). In this case, the aberration should be applied before the final least-squares solution for the single FOV.

We have chosen to correct for aberration when a sub-catalog is selected from the mission catalog. This decision was based on programing ease although it does require more time to correct the whole sample rather than only the matched stars. The impact is not severe, however, since the calculation is very simple.

Appendix 6: Least-Squares Correction Techniques

In Process B we seek to minimize the sum of the squares of the residuals between measured star image coordinates and predicted coordinates for the same stars, using direction cosines from the on-board catalog. The mapping of catalog positions onto the CCD image plane is a function of Euler parameters via the stellar colinearity equations:

$$x = f \left\{ \frac{AN_{11}L_1 + AN_{12}L_2 + AN_{13}L_3}{AN_{31}L_1 + AN_{32}L_2 + AN_{33}L_3} \right\} \quad (A6.1)$$

$$y = f \left\{ \frac{AN_{21}L_1 + AN_{22}L_2 + AN_{23}L_3}{AN_{31}L_1 + AN_{32}L_2 + AN_{33}L_3} \right\}$$

where

f = lens focal length

AN_{ij} = elements of the coordinate frame rotation matrix $[AN]$.

L_i = star direction cosines for the particular star, measured in the N frame.

If we let:

$X = \{(x_i, y_i)\}$ = vector of calculated CCD image plane coordinates.

$\tilde{X} = \{(x_i, y_i)\}_m$ = vector of measured star image positions on the CCD.

and

$\Delta X = \tilde{X} - X$ = vector of residuals,

then we seek to find the set of Euler parameters, β , such that the weighted sum of the squares of the residuals is minimized; i.e., minimize

$$\phi = \Delta X_p^T W \Delta X_p \quad (A6.2)$$

where

$$\Delta X_p = \tilde{X} - X_p$$

and

X_p = vector of linearly predicted image coordinates.

But, by first-order Taylor expansion

$$\Delta X_p = \Delta X_c - A\Delta\beta$$

where

ΔX_c = vector of current image coordinate residuals based current estimates of β .

A = matrix of partial derivatives of the colinearity equations with respect to Euler parameters.

$\Delta\beta$ = corrections to the current estimates of Euler parameters.

Thus, we can write:

$$\phi_p = (\Delta X_c - A\Delta\beta)^T W(\Delta X_c - A\Delta\beta). \quad (A6.4)$$

In addition to finding the set of Euler parameters to minimize ϕ_p , we must also satisfy the constraint equation:

$$\beta^T \beta = 1. \quad (A6.5)$$

Letting $\beta_p = \beta_c + \Delta\beta$, we find:

$$(\beta_c + \Delta\beta)^T (\beta_c + \Delta\beta) = 1$$

or to first order:

$$1 - \beta_c^T \beta_c = 2\beta_c^T \Delta\beta. \quad (A6.6)$$

Thus, our problem requires that we minimize Eq. A6.4 subject to the constraint equation, Eq. A6.6. The constraint equation can be incorporated in Eq. A6.4 as an additional *perfect* observation equation but with a large weight. That is, $\Delta Y = (1 - \beta_c^T \beta_c)$ is appended to the ΔX_c vector and $2\beta_c^T$ appended as an additional row into the A matrix. The relative weight

for this equation, the last element of W , is chosen large enough (about 10^3) so that $(A^T W A)^{-1}$ does not change appreciably for variations in this weight. Then, for minimization, we require:

$$\nabla_{\Delta\beta} \phi = -2A^T W \Delta X_C + 2(A^T W A) \Delta\beta = 0$$

or

$$\Delta\beta = (A^T W A)^{-1} A^T W \Delta X_C. \quad (A6.7)$$

Determination of Interlock Euler Parameters Between FOV(A) and FOV(B)

Process B, in analyzing star image data, first treats FOV(A) and FOV(B) independently. The least-squares differential correction determines the best estimate of the Euler parameters (β_{VN}) orienting the vehicle frame, V (see Appendix 7), relative to the inertial frame, N . For FOV(B) the interlock relationship between FOV(A) and FOV(B) (β_{BA}) is assumed known and β_{VN} is adjusted again. In reality, however, the interlocks do vary slightly with time. Therefore, we have expanded our algorithm to treat the combined data from FOV(A) and FOV(B) in order to determine, simultaneously, the β_{VN} and β_{BA} which minimize (in a least-squares sense) the star image coordinate residuals (see Appendix 7 for further details).

In order to rigorously interpret $(A^T W A)^{-1}$ as the 8×8 covariance matrix of the estimated Euler parameters, W should be chosen as the inverse of the "measurement" covariance matrix. However, since a scale factor on W is formally immaterial in the least squares solution and assuming all measurement errors are uncorrelated we exercise the simple option of setting W to an identity matrix except for the larger constraint weights; the correct covariance matrix is obtained by simply multiplying the converged $(A^T W A)^{-1}$ matrix by the image coordinate measurement variance. The two constraints of the form (A6.6) are treated as "perfect

measurements". Thus, it is clear that the two corresponding formal weights are ∞ ; it is equally clear that we are limited to choosing a sufficiently large number (about 10^3) in practice. We have carried out sufficient experimentation to expect no implementation problems here; the differential correction process converges well.

We have found it desirable to further process the interlock Euler parameters, β_{BA} . We assume (justifiably!) that these interlocks vary slowly with time, but the interlocks calculated by the combined least-squares method, discussed above, show relatively large scatter about their true values. This sensitivity is due to the relatively poor determination of the roll about each boresight (this does not affect the determination of β_{VN}). To better monitor the interlock parameters we adopted a discrete Kalman filter algorithm to combine the predicted β_{BA} , determined from previous data frames, with the β_{BA} computed by the least-squares correction. The equations needed for this are:

$$\begin{aligned}\hat{\beta}_{BA}(k) &= \bar{\beta}_{BA}(k) + K(k)(\tilde{\beta}_{BA}(k) - \bar{\beta}_{BA}(k)) \\ K(k) &= P_{k-1}(k)(L_{V_k}V_k + P_{k-1}(k))^{-1} \\ P_{k-1}(k) &= P_{k-1}(k-1) + Q(k) \\ P_k(k) &= (I - K(k))P_{k-1}(k) \\ Q(k) &= BQ'B^T \\ B &= \frac{\partial \beta_{BA}}{\partial \phi}\end{aligned}\tag{A6.7a-d}$$

where $\hat{\beta}_{BA}(k)$ = optimal estimate of interlock parameters at time t_k
obtained by combining predicted and computed values,

- $\bar{\beta}_{BA}(k)$ = predicted interlock parameters at time t_k based on previous analysis,
 $\tilde{\beta}_{BA}(k)$ = interlock parameters at time t_k obtained via least-squares correction,
 $K(k)$ = Kalman gain matrix at time t_k ,
 $P_i(j)$ = covariance matrix associated with β_{BA} at time t_j based on analysis of interlock values through time t_i ,
 $L_{V_k} V_k$ = covariance matrix associated with calculated interlock parameters, $\tilde{\beta}_{BA}(k)$,
 $Q(k)$ = process noise matrix,
 Q' = process noise matrix for 3 interlock angles ($\phi = \{\phi_1, \phi_2, \phi_3\}$)

Formally, $P_{k-1}(k)$ would be obtained by forward integration of the matrix Riccati equation, or by other methods. However, since to first order, $\dot{\beta}_{BA} = 0$ and assuming the process noise is constant in time as well, $P_{k-1}(k)$ is obtained by simply adding the time integral of the process noise, $Q(k)$, to the previous covariance, $P_{k-1}(k-1)$. In addition, for simplicity we assume Q' is diagonal with equal noise for each angle. For programming ease we have used a pre-calculated B matrix, valid for our nominal interlock arrangement.

The process noise matrix Q' essentially controls the scatter of the estimated Euler parameters. Small values for the elements of Q' will permit little change in the $\bar{\beta}_{BA}$ (making it insensitive to each new $\tilde{\beta}_{BA}$). However, this may cause the $\hat{\beta}_{BA}$ not to "track" the true variations. Conversely, large values for Q' elements will cause more scatter in $\hat{\beta}_{BA}$. Obviously, Q' is a "tuning" parameter which must be selected for the particular system.

Appendix 7: Orientation of the Vehicle Frame

We describe the vehicle orientation by the set of Euler parameters, β_{VN} , which orient a "V" frame (defined below) relative to the inertial frame, "N". The V frame is defined entirely by the boresight vectors of the two star sensors [FOV(A), FOV(B)]. Given the boresight of FOV(A) as \underline{a}_3 and of FOV(B) by \underline{b}_3 , we define the V frame unit vectors as follows:

$$\begin{aligned}\underline{v}_1 &= (\underline{a}_3 + \underline{b}_3) / (|\underline{a}_3 + \underline{b}_3|) \\ \underline{v}_3 &= (\underline{a}_3 \times \underline{b}_3) / (|\underline{a}_3 \times \underline{b}_3|) \\ \underline{v}_2 &= \underline{v}_3 \times \underline{v}_1\end{aligned}\tag{A7.1}$$

The advantage of this frame is that since the boresights of the two FOV are well determined so also will be the V frame and, hence, the β_{VN} parameter set. The poorly determined roll angle about each boresight will not affect β_{VN} .

In addition to β_{VN} , we also make use of β_{BA} , the Euler parameters orienting FOV(B) with respect to FOV(A). These parameters are monitored as a means for monitoring the interlock angles between FOV(A) and FOV(B).

As we will show below, we do not actually need the \underline{v} unit vectors calculated by the above equations. We do need, however, the rotation matrix AV which rotates the V frame into the FOV(A) frame. The matrix AV can be calculated from the BA rotation matrix.

Matrix AV can be constructed by filling its columns with the \underline{v} unit vectors expressed in the A frame. We first express \underline{b}_3 in terms of \underline{a} unit vectors in order to calculate the vectors \underline{v}_1 , \underline{v}_2 , \underline{v}_3 (in the A frame).

$$\underline{b}_3^A = BA \underline{a}_3^A\tag{A7.2}$$

And since

$$\underline{a}_3^A = \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix} \quad (A7.3)$$

we see that

$$\underline{b}_3^A = \begin{Bmatrix} BA_{31} \\ BA_{32} \\ BA_{33} \end{Bmatrix}. \quad (A7.4)$$

Therefore, in the A frame:

$$\begin{aligned} \underline{v}_1 &= (\underline{a}_3 + \underline{b}_3) / (|\underline{a}_3 + \underline{b}_3|) \\ &= \left\{ \frac{BA_{31}}{(2 + 2BA_{33})^{1/2}}, \frac{BA_{32}}{(2 + 2BA_{33})^{1/2}}, \frac{1 + BA_{33}}{(2 + 2BA_{33})^{1/2}} \right\}^T \end{aligned} \quad (A7.5)$$

$$\begin{aligned} \underline{v}_3 &= (\underline{a}_3 \times \underline{b}_3) / (|\underline{a}_3 \times \underline{b}_3|) \\ &= \left\{ \frac{-BA_{32}}{(1 - BA_{33}^2)^{1/2}}, \frac{BA_{31}}{(1 - BA_{33}^2)^{1/2}}, 0 \right\}^T \end{aligned} \quad (A7.6)$$

$$\begin{aligned} \underline{v}_2 &= \underline{v}_3 \times \underline{v}_1 \\ &= \left\{ \frac{BA_{31}}{(2 - 2BA_{33})^{1/2}}, \frac{BA_{32}}{(2 - 2BA_{33})^{1/2}}, \frac{-(2 - 2BA_{33})^{1/2}}{2} \right\}^T \end{aligned} \quad (A7.7)$$

The vectors, \underline{v}_i , expressed in the A frame, form the columns of matrix AV. We see that AV is a function of just 3 variables BA_{3i} , $i = 1, 2, 3$. For the least-squares differential correction of Process B we need the partial derivatives $\partial AV / \partial \beta_j$, $j = 0, 1, 2, 3$. To simplify the calculations we expand this:

$$\frac{\partial AV}{\partial \beta_j} = \sum_{i=1}^3 \frac{\partial AV}{\partial BA_{3i}} \frac{\partial BA_{3i}}{\partial \beta_j} \quad (A7.8)$$

where $\frac{\partial AV}{\partial BA_{3i}}$ is a 3×3 matrix and $\frac{\partial BA_{3i}}{\partial \beta_j}$ are elements of a 3×3 matrix $\frac{\partial BA}{\partial \beta_j}$. The partials $\partial AV / \partial BA_{3i}$, after some algebra, are

$$\begin{aligned} \frac{\partial AV}{\partial BA_{31}} &= \begin{bmatrix} \frac{1}{(2 + 2BA_{33})^{1/2}} & \frac{1}{(2 - 2BA_{33})^{1/2}} & 0 \\ 0 & 0 & \frac{1}{(1 - BA_{33}^2)^{1/2}} \\ 0 & 0 & 0 \end{bmatrix} \\ \frac{\partial AV}{\partial BA_{32}} &= \begin{bmatrix} 0 & 0 & \frac{-1}{(1 - BA_{33}^2)^{1/2}} \\ \frac{1}{(2 + 2BA_{33})^{1/2}} & \frac{1}{(2 - 2BA_{33})^{1/2}} & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \frac{\partial AV}{\partial BA_{33}} &= \begin{bmatrix} \frac{-BA_{31}}{(2 + 2BA_{33})^{3/2}} & \frac{BA_{31}}{(2 - 2BA_{33})^{3/2}} & \frac{-BA_{32}BA_{33}}{(1 - BA_{33}^2)^{3/2}} \\ \frac{-BA_{32}}{(2 + 2BA_{33})^{3/2}} & \frac{BA_{32}}{(2 - 2BA_{33})^{3/2}} & \frac{BA_{31}BA_{33}}{(1 - BA_{33}^2)^{3/2}} \\ \frac{1}{2(2 + 2BA_{33})^{1/2}} & \frac{1}{2(2 - 2BA_{33})^{1/2}} & 0 \end{bmatrix} \end{aligned} \quad (A7.9a-c)$$

We now consider in some detail how the derivative matrix for the least-squares differential correction is filled to recover β_{VN} and β_{BA} . The

elements of the A matrix for treating a single FOV are found by expanding

$$\frac{\partial X}{\partial \beta_{VN}}:$$

$$\begin{aligned} \left. \frac{\partial X}{\partial \beta_{VN}} \right|_A &= \frac{\partial X}{\partial AN} \frac{\partial AN}{\partial \beta_{VN}} \quad [\text{for FOV(A)}] \\ &= \frac{\partial X}{\partial AN} \left(\frac{\partial AV}{\partial \beta_{VN}} VN + AV \frac{\partial VN}{\partial \beta_{VN}} \right) \quad (\text{using } AN = AV \cdot VN) \\ &= \frac{\partial X}{\partial AN} \left(AV \frac{\partial VN}{\partial \beta_{VN}} \right) \end{aligned} \quad (A7.10)$$

$$\begin{aligned} \left. \frac{\partial X}{\partial \beta_{VN}} \right|_B &= \frac{\partial X}{\partial BN} \frac{\partial BN}{\partial \beta_{VN}} \quad [\text{for FOV(B)}] \\ &= \frac{\partial X}{\partial BN} \left(\frac{\partial BV}{\partial \beta_{VN}} VN + BV \frac{\partial VN}{\partial \beta_{VN}} \right) \quad (\text{using } BN = BV \cdot VN \text{ and } BV = BA \cdot AV) \\ &= \frac{\partial X}{\partial BN} \left(BV \frac{\partial VN}{\partial \beta_{VN}} \right). \end{aligned} \quad (A7.11)$$

After matching at least 3 stars in each FOV, we can combine both FOV to recover the interlock parameter β_{BA} as well as β_{VN} . In this case, we need the following matrix derivatives:

$$\begin{aligned} \left. \frac{\partial X}{\partial \beta_{VN}} \right|_A &= \frac{\partial X}{\partial AN} \left(AV \frac{\partial VN}{\partial \beta_{VN}} \right) \\ \left. \frac{\partial X}{\partial \beta_{BA}} \right|_A &= \frac{\partial X}{\partial AN} \left(\frac{\partial AV}{\partial \beta_{BA}} VN + AV \frac{\partial VN}{\partial \beta_{BA}} \right) = \frac{\partial X}{\partial AN} \left(\frac{\partial AV}{\partial \beta_{BA}} \frac{\partial BA}{\partial \beta_{BA}} VN \right) \quad (A7.12) \\ \left. \frac{\partial X}{\partial \beta_{VN}} \right|_B &= \frac{\partial X}{\partial BN} \left(BV \frac{\partial VN}{\partial \beta_{VN}} \right) \\ \left. \frac{\partial X}{\partial \beta_{BA}} \right|_B &= \frac{\partial X}{\partial BN} \left(\frac{\partial BA}{\partial \beta_{BA}} AN + BA \frac{\partial AV}{\partial \beta_{BA}} VN + BA \cdot AV \frac{\partial VN}{\partial \beta_{BA}} \right) \end{aligned}$$

$$\left. \frac{\partial X}{\partial \beta_{BA}} \right|_B = \frac{\partial X}{\partial \beta_N} \frac{\partial \beta_A}{\partial \beta_{BA}} A_N + \beta_A \frac{\partial A_V}{\partial \beta_A} \cdot \frac{\partial \beta_A}{\partial \beta_{BA}} \quad (A7.13)$$

The elements of the vectors ΔX and $\Delta \beta$ and matrix A are:

$$\Delta X = \begin{bmatrix} (\Delta x, \Delta y)_A & = \text{vector of image residuals for FOV(A)} \\ (\Delta x, \Delta y)_B & = \text{vector of image residuals for FOV(B)} \\ 1 - \beta_{VN}^T \beta_{VN} & = \text{constraint condition for } \beta_{VN} \\ 1 - \beta_{BA}^T \beta_{BA} & = \text{constraint condition for } \beta_{BA} \end{bmatrix}$$

$$\Delta \beta = \begin{bmatrix} \Delta \beta_{VN} = \text{correction vector for } \beta_{VN} \\ \Delta \beta_{BA} = \text{correction vector for } \beta_{BA} \end{bmatrix}$$

$$A = \begin{bmatrix} \left. \frac{\partial(x, y)}{\partial \beta_{VN}} \right|_A & \left. \frac{\partial(x, y)}{\partial \beta_{BA}} \right|_A \\ \left. \frac{\partial(x, y)}{\partial \beta_{VN}} \right|_B & \left. \frac{\partial(x, y)}{\partial \beta_{BA}} \right|_B \\ \beta_{VN}^T W_1^{1/2} & 0 \\ 0 & \beta_{BA}^T W_1^{1/2} \end{bmatrix}$$

Then $\Delta \beta = (A^T W A)^{-1} A^T W \Delta X$; the corrections are added to β_{VN} and β_{BA} and the process is repeated until $\Delta \beta$ is small. The final values for β_{VN} are passed to Process C.

Appendix 8: Riccati Equation Covariance Propagation

The Kalman filter formulated for Process C includes a direct numerical integration of the covariance matrix between two time points. This allows a more rigorous incorporation of the process noise component in the covariance matrix propagation. Our initial method was to compute the state transition matrix to use this to propagate covariance by pre- and post-multiplication of the previous covariance matrix. An estimate of the process noise estimate was then added on. We outline the covariance integration technique below.

We have chosen as our state vector:

$$\underline{X} = \begin{bmatrix} \underline{X_1} \\ \underline{X_2} \end{bmatrix} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ -- \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 4 \text{ Euler parameters, } \beta_{VN}, \text{ orienting the} \\ \text{vehicle frame with respect to inertial} \\ \text{frame, N, and 3 gyro bias values, } \underline{b}. \end{bmatrix}$$

The state differential equations for our system are:

$$\begin{aligned} \{\dot{X}_1\} &= \{\dot{\beta}_{VN}\} = [\beta_{VN}][VG]\{\omega_{GN}\} \\ &= [\beta_{VN}]\{\omega_{VN}\} \end{aligned} \quad (A8.1a)$$

$$= [\omega_{VN}]\{\beta_{VN}\} \quad (A8.1b)$$

$$\{\dot{X}_2\} = \{\dot{b}\} = 0 \quad (A8.2)$$

where

$$[\beta] = \frac{1}{2} \begin{bmatrix} -\beta_1 & -\beta_2 & -\beta_3 \\ \beta_0 & -\beta_3 & \beta_2 \\ \beta_3 & \beta_0 & -\beta_1 \\ -\beta_2 & \beta_1 & \beta_0 \end{bmatrix}$$

$$[\omega] = \frac{1}{2} \begin{bmatrix} 0 & -\omega_1 & -\omega_2 & -\omega_3 \\ \omega_1 & 0 & \omega_3 & -\omega_2 \\ \omega_2 & -\omega_3 & 0 & \omega_1 \\ \omega_3 & \omega_2 & -\omega_1 & 0 \end{bmatrix}$$

$\{\omega_{GN}\}$ = vector of true rotation rates of the vehicle
about three orthogonal body fixed axes,

$[VG]$ = 3×3 rotation matrix which rotates the gyro
rates from the gyro frame to the vehicle frame.

Our model for each gyro measurement, $\tilde{\omega}_{GN}$, includes an unknown noise term, V , and a gyro bias, b_{GN} , which we can estimate. Then, the state differential equation for $\{X_1\}$ becomes:

$$\begin{aligned} \{\dot{X}_1\} &= [\beta_{VN}][VG](\{\tilde{\omega}_{GN}\} - \{b_{GN}\} + \{V\}) \\ &= [\tilde{\omega}_{VN}]\{\beta_{VN}\} - [\beta_{VN}][VG]\{b_{GN}\} + [\beta_{VN}][VG]\{V\}. \end{aligned} \quad (A8.3)$$

Combining the two state differential equations, they can be rewritten in a linear form:

$$\dot{\underline{X}} = \underline{F}\underline{X} + \underline{G}\underline{V},$$

where

$$\underline{F} = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix}$$

$$F = \begin{bmatrix} [\tilde{\omega}_{VN}] & -[\beta_{VN}][VG] \\ 0 & 0 \end{bmatrix}$$

$$G = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} = \begin{bmatrix} [\beta_{VN}][VG] & 0 \\ 0 & S*[I] \end{bmatrix}$$

S = a scale factor or "tuning" parameter.

The covariance matrix propagation is calculated by numerical integration of the matrix Riccati equation:

$$\dot{P} = FP + PF^T + GQG^T \quad (A8.4)$$

where P is the covariance matrix and Q is the process noise covariance matrix which, in our case, represents a measure of the noise covariance between gyro rates about the three axes and between the three gyro biases. This matrix is taken to be a 6 x 6 diagonal matrix. In order to speed computation we partition this equation:

$$\begin{bmatrix} \dot{P}_{11} & \dot{P}_{12} \\ \dot{P}_{21} & \dot{P}_{22} \end{bmatrix} = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} + \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \begin{bmatrix} F_{11}^T & F_{21}^T \\ F_{12}^T & F_{22}^T \end{bmatrix} + \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \begin{bmatrix} G_{11}^T & G_{21}^T \\ G_{12}^T & G_{22}^T \end{bmatrix}$$

Since $F_{21} = 0$, $F_{22} = 0$, $G_{12} = 0$, $G_{21} = 0$, $Q_{12} = 0$, and $Q_{21} = 0$ we can write the set of four equations implied in the above equation as

$$\dot{P}_{11} = F_{11}P_{11} + F_{12}P_{21} + P_{11}F_{11}^T + P_{12}F_{12}^T + G_{11}Q_{11}G_{12}^T$$

$$\dot{P}_{12} = F_{11}P_{12} + F_{12}P_{22}$$

$$\dot{P}_{21} = P_{21}F_{11}^T + P_{22}F_{12}^T$$

$$\dot{P}_{22} = G_{22}Q_{22}G_{22}^T.$$

Also, since P is symmetric, $P_{12} = P_{21}^T$, $P_{11} = P_{11}^T$, $P_{22} = P_{22}^T$ and since $F_{11} = [\omega]$, which is skew symmetric, $F_{11}^T = -F_{11}$, and $G_1 = -F_{12}$. Therefore,

$$\dot{P}_{11} = (F_{11}P_{11} + F_{12}P_{21}) + (F_{11}P_{11} + F_{12}P_{21})^T + F_{12}QF_{12}^T$$

$$\dot{P}_{21} = P_{22}F_{12}^T - P_{21}F_{11} = \dot{P}_{12}^T.$$

We must integrate three matrix differential equations, \dot{P}_{11} , \dot{P}_{21} and \dot{P}_{22} to propagate the covariance matrix. Only two matrices need to be filled at each time step in the integration: (\dot{P}_{22} is a constant)

$$F_{11} = [\dot{\omega}_{VN}]$$

and

$$F_{12} = -[\beta][VG].$$

The simplest form for the state differential equation for the β_{VN} is then

$$\dot{\beta} = [F_{12}](\{b_{GN}\} - \{\dot{\omega}_{GN}\})$$

Currently, we are using a two cycle Runge-Kutta numerical integration for the Riccati equation.

Kalman Filter State Update with A-Priori Information

As mentioned in the Phase II report, the Kalman filter, as formulated in that report, did not estimate the gyro bias values with the desired precision. This shortcoming was due, in part, to the gyro rate noise

having a magnitude similar to the biases themselves. We have remedied this problem by reformulating the Kalman filter to treat the biases as observable with an associated covariance matrix. Such a method seems justified since, in general, the biases vary slowly on the time scale of minutes and, therefore, we would have some knowledge of them obtained from previous iterations.

The general form for the discrete Kalman filter state update equation for a linear system is

$$\hat{\underline{X}}_{k+1}(k+1) = \underline{\bar{X}}_k(k+1) + K(k+1)[\underline{Z}(k+1) - H(k+1)\underline{\bar{X}}_k(k+1)] \quad (\text{A8.5})$$

where \underline{Z} is the measurement vector at time t_{k+1} and $\underline{\bar{X}}_k(k+1)$ is the state, integrated from time t_k to t_{k+1} . In our formulations, \underline{Z} contains the output of Process B, the set of Euler parameters β_{VN} relating the V frame to the inertial frame, and the three bias values from the previous pass through the Kalman filter. Thus, the observation vector \underline{Z} contains the same variables as the state vector making matrix $H = [I]$.

Looking now at the other equations for the Kalman filter:

$$\begin{aligned} K(k+1) &= P_k(k+1)H^T(k+1)[L(k+1) + H(k+1)P_k(k+1)H^T(k+1)]^{-1} \\ &= P_k(k+1)[L(k+1) + P_k(k+1)]^{-1} \end{aligned} \quad (\text{A8.6})$$

where $L(k+1)$ = covariance matrix for the measurements and $P_k(k+1)$ = integrated covariance from the Riccati equation. The upper left 4×4 portion of L is the covariance matrix from the least-squares results of Process B. The lower right 3×3 portion of L is the covariance matrix of the gyro biases. This we have assumed to be diagonal (gyro biases are assumed independent); off diagonal portions of L are assumed to be zero.

The covariance matrix update equation is

$$\begin{aligned}
 P_{k+1}(k+1) &= [I - K(k+1)H(k+1)]P_k(k+1) \\
 &= P_k(k+1) - K(k+1)P_k(k+1)
 \end{aligned}
 \tag{A8.7}$$

Bias Estimation Results

Our simulations indicate that the formulation discussed above functions extremely well. We note that as the estimated variance is decreased we get an improvement in the bias estimate. However, a stricter value on the bias variance also decreases the response time of the system in tracking a bias change.

It must be kept in mind that the gyro biases, as we have defined them, include not only true gyro bias but also other effects such as gyro nonorthogonality, gravity or magnetic effects, and poorly known interlock angles between sensor and gyro frames. It is expected that on the short term the gyro bias terms will absorb these *errors* and permit the state integrations to yield good updates.

Appendix 9: Rate Gyro Readout Data Generation

The simplest or nominal rotation history for a satellite for our study is one rotation per revolution around the earth. For our geometry this would be rotation about the g_2 unit vector, normal to the orbit plane. To be more realistic and provide a challenge to the software, we have formulated a more complicated read-out record for the gyroscopes. In a satellite there may be various motors, panels and antennae. Each of these may vibrate at various frequencies. Therefore, we have generated gyro rates with a spectrum of frequencies, phases and amplitudes. To do this we have created data in the (Discrete) Fourier Transform space or frequency space. A Fast Fourier Transform of the data yields a gyro record. We used the formula

$$f_j = e^{-10j} G_j \quad (A9.1)$$

to generate a frequency spectrum, where G_j is a Gaussian distributed random number. We make the real transform symmetric about zero frequency and the imaginary data anti-symmetric in order yield a real gyro record. With proper scaling, we have used the above form to generate a gyro readout every 0.5 sec. for each gyro axis.

Proper application of the above technique yields a realistic gyro record, but it has the disadvantage of making it difficult to specify a *true* rotation and attitude history for the spacecraft. That is, Runge-Kutta integration is designed to integrate smooth functions whereas this rate simulation is discrete. Also, in a real system some of the frequencies recorded by the gyroscopes are just vibration or noise and not rotation. For consistency, we decided to integrate the gyro record using two-cycle

Runge-Kutta in both the truth model programs and Process C. This at least allows comparison between models with and without additional noise. We have performed several tests to determine the effects of gyro noise and a complicated signal. These tests confirm our intuition that if we take a sufficient number of samples, the rapid zero mean oscillations about the mean motion do not significantly affect the integrated solution.

Appendix 10: Software Documentation, Program
Listings and Sample Output

This appendix is divided into several parts. We first describe and list the program which generates simulated data, Datgen, and the program which processes the data using Processes B and C, Combin. Then, since these two programs have several subroutines in common, we describe and list all the subroutines for these programs, in alphabetical order. Figures A10.1 and A10.2 display the hierarchy of subroutine calls for these two programs.

A sample of program output is presented also.

AD-A103 806 VIRGINIA POLYTECHNIC INST AND STATE UNIV BLACKSBURG --ETC F/6 17/7
STAR PATTERN RECOGNITION AND SPACECRAFT ATTITUDE DETERMINATION.(U)
MAY 81 T E STRIKWERDA, J L JUNKINS DAAK70-78-C-0038

UNCLASSIFIED

ETL-0260

NL

2 OF 3

AL-4

OSBOR

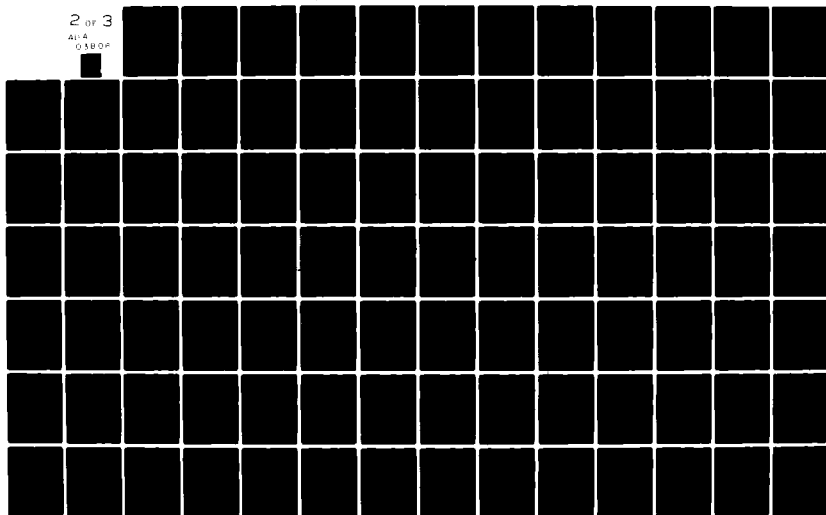


Table of Contents for Appendix 10.

	<u>Page</u>
<u>Main Programs</u>	
Datgen Description	93
Datgen Listing	99
Combin Description	109
Combin Listing	112
<u>Subroutines</u>	
Access	118
Bias	121
Cross	123
Deriv	125
Circosb	127
Dxdbeta	130
Fill-y	133

<u>Subroutine</u>	<u>Page</u>
Gauss	135
Kalman	137
Least	140
Least-1	142
Least-2	145
Mata	149
Mat-av	152
Orbit	155
Pairit	158
Perturb	164
Phoegn	166
Post-mult	168
Pre-mult	170
Proc-b	172

<u>Subroutine</u>	<u>Page</u>
Runge	177
Secant	181
Sort	184
Sample Output from Datgen	186
Sample Output from Combin (Program start)	189
Sample Output from Combin (frame 5)	190

Datgen

This program generates simulated data which is used for tests of Processes B and C. The data for a sequence of frames are written to a disk or tape file for later analysis. For each simulation test we can select variations in any of a number of parameters. However, the program itself must be modified to change the amplitude, period or form of the variation. To run this program it is necessary to supply a file containing realistic gyroscope read-out rates (currently, enough for 15 minutes of satellite motion with a spacing of 0.5 seconds between readouts), and a mission catalog of star positions (ordered into cells by the method of Appendix 3).

To begin generating data, the gyro rates and the table of star catalog cell positions are read from the external files. Next, we select the variations we wish to include in this simulation. The earth and satellite orbit constants and initial positions and velocities are specified in the program but can be changed if desired. These orbit parameters, along with our assumed geometry, are used to determine the initial orientation of the vehicle via rotation matrix V_N , orienting the vehicle frame relative to the inertial frame; from V_N we recover the initial values of Euler parameters, β_{VN} . All constants for this simulation are stored in the first record of the data file (see Table A10.1).

We are now ready to compute image coordinates for successive frames of data (separated by 30 seconds of satellite motion in the present software version). We note that all variations such as gyro biases and coordinate frame perturbations are slowly varying and, therefore,

the perturbed values are calculated only once per 30 second interval (subroutines Bias and Perturb). For each frame of data we first integrate forward (via subroutine Runge) the kinematic equation for β_{VN} using the "true" gyro rates, perhaps including a perturbed rotation matrix VG , found from β_{VG} and used to rotate the gyro rates from the gyro frame into the vehicle frame. Gaussian noise is added to the integrated values of β_{VN} in order to provide an "estimate" of β_{VN} to start Process B, if it is run separately from Process C (see description of Combin). Euler parameters β_{BA} which can be time varying and rotation matrix BA , relating FOV(B) with respect to FOV(A), are determined at this time also. We then compute the position and velocity vectors of the earth and satellite (with subroutine Orbit) and find the total velocity (which is later used by subroutine Access to add stellar aberration to star direction cosines). The various Euler parameters and total velocity are saved in the first part of each data record.

The next step is to calculate image coordinates for stars in each FOV. We calculate rotation matrix AN (or BN for FOV(B)) using subroutines Dircosb and Mat-av, and use the last row as the boresight unit vector of the star tracker. This unit vector is used by Access to retrieve a subcatalog of stars. Each star is then projected onto the focal plane (via Phoeqn); if it lies within the CCD border, we save its position and magnitude. These "true" positions are then perturbed with Gaussian noise (subroutine Gauss) to produce "measured" star coordinates (up to 10 stars). The steps outlined above are repeated for FOV(B). Both the true and measured image coordinates and magnitudes are saved on the data file.

The last step in each frame is to add noise and/or bias values to the rate gyro data (for that frame) and save the results on the data file (see Table A10.2). Sufficient space is left between the end of this information and the end of the record so results of the analysis of Processes B and C can be saved for later evaluation.

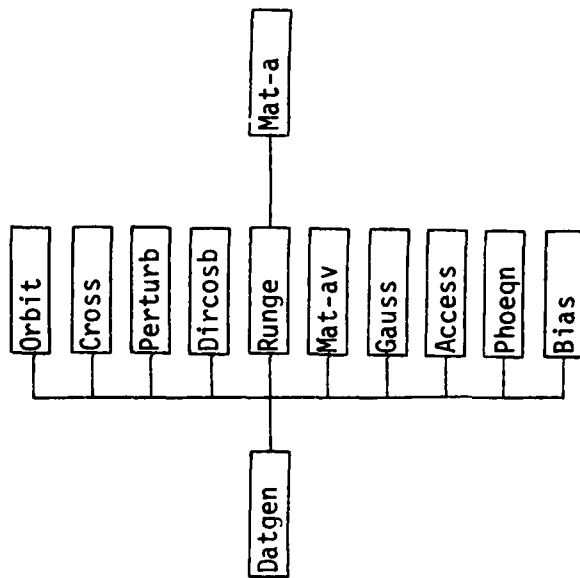


Figure A10.1: Hierarchy of subroutine calls (left to right) for simulation program Datgen.

Table A10.1: Format of First Record of Simulation File

<u>Variable</u>	<u>Size</u>	<u>Locations</u>
Satellite orbit major axis	1	1
G*(mass of earth)	1	2
Initial satellite position	3	3-5
Initial satellite velocity	3	6-8
Earth orbit major axis	1	9
G*(mass of sun)	1	10
Initial earth position	3	11-13
Initial earth velocity	3	14-16
Satellite orbit inclination	1	17
Image centroid error (1-sigma)	1	18
Image magnitude error (1-sigma)	1	19
Euler parameter error (1-sigma)	1	20
Gyro rate error (1-sigma)	1	21
Gyro read-out spacing	1	22
Starting time (seconds)	1	23
Frame spacing (seconds)	1	24
Nominal interlock Euler parameters β_{BA}	4	25-28
Variation amplitudes	4	29-32
Variation frequency factors	4	33-36
Nominal gyro biases	3	37-39
Variation amplitudes	3	40-42
Variation frequency factors	3	43-45
Nominal interlock Euler parameters β_{VG}	4	46-49
Variation amplitudes	4	50-53
Variation frequency factors	4	54-57
Weight for apriori Euler parameters β_{BA}	1	58
Standard deviation for gyro biases	1	59

Table A10.2: Format of Data Records of Simulation File

	Variable	Size	Bytes/ Number	Location
Simulation Program Output Data	β_{VN} (true)	4	8	1-4
	β_{VN} (estimated)	4	8	5-8
	β_{VG} (true)	4	8	9-12
	β_{VG} (estimated)	4	8	13-16
	β_{BA} (true)	4	8	17-20
	β_{BA} (estimated)	4	8	21-24
	Velocity components	3	8	25-27
	No. stars in FOV(A)	1	8	28
	$\{(x,y,m)\}$ (true)	10×3	4	29-58
	$\{(x,y,m)\}$ (measured)	10×3	4	59-88
	No. stars in FOV(B)	1	8	89
	$\{(x,y,m)\}$ (true)	10×3	4	90-119
	$\{(x,y,m)\}$ (measured)	10×3	4	120-149
	Rate gyro data (ω_1)	61	4	150-210
	Rate gyro data (ω_2)	61	4	211-271
	Rate gyro data (ω_3)	61	4	272-332
	Gyro biases (true)	3	8	333-335
Process B Output Data	β_{VN} (calc.)	4	8	336-339
	β_{BA} (calc.)	4	8	340-343
	Covariance matrix for β_{VN} (calc.)	4×4	4	344-359
	No. stars matched in FOV(A)	1	8	360
	$\{(x,y)\}$ (calc.)	5×2	4	361-370
	No. stars matched in FOV(B)	1	8	371
Process C Output Data	$\{(x,y)\}$ (calc.)	5×2	4	372-381
	β_{VN} (optimal estimate)	4	8	382-385
	Gyro biases (optimal estimate)	3	8	386-388
	β_{VN} (integrated from previous value)	4	8	389-392
	Covariance matrix for β_{VN} (opt. est.)	4×4	4	393-408

```
*****
*
*           P R O G R A M : D A T G E N
*
*****
```

```
80      ! This program computes data for Processes B and C. Data consist of Euler
90      ! parameters, assorted other parameters, and image coordinates for
100     ! both fields of view.
110     ! We use Euler parameters in this version to orient
120     ! the V frame relative to inertial frame and FOV(B) w.r.t. FOV(A).
130     ! Tom Strickwerca...V.P.I. & S.U.....14 JANUARY 1981.
140     !
150     OVERLAP
160     OPTION BASE 1
170     FLOAT %
180     DIM An(3,3),Gn(3,3),Ba(3,3),Bn(3,3),Av(3,3),Vg(3,3),Vn(3,3)
190     DIM G1(3),G2(3),G3(3),Bore(3)
200     DIM Ps0(3),Vs0(3),Ps(3),Vs(3),L(3),Lp(3),Bias(3)
210     DIM Pe0(3),Ve0(3),Pe(3),Ve(3),V(3),Vu(3),Xn(3),Voc(3)
220     DIM Bun(4),Bonest(4),Bug(4),Bba(4)
230     SHORT Xym(10,3),Xyt(10,3),Fov(100,4)
240     SHORT Bbanom(4),Biasnom(3),Gyronom(3),Bugnom(4)
250     SHORT Eug(4),Nug(4),Eba(4),Nba(4),Ebias(3),Nbias(3),Egyro(3),Ngyro(3)
260     SHORT W1(2048),W2(2048),W3(2048),R1(61),R2(61),R3(61)
270     INTEGER Table(529,2)
280     RANDOMIZE
290     !
300     Wread=0
310 Restart:      ! Come here to do another run.
320     PRINT USING "K";"***** P R O G R A M D A T G E N *****"
330     !
340     N$="N"
350     INPUT "Do you want to use realistic gyro rate history (Y/N)?",N$
360     PRINT USING "/K,A";"Do you want to use realistic gyro rate history (Y/N)?";N$
370     !
380     IF N$<>"Y" THEN Plain_rates
390     IF Wread=1 THEN Star_cat
400     PRINT USING "/K";"Place disk with gyro rates (Filename:'Wtrue') in :F8,1..
...then push CONT."
410     PAUSE
420     ASSIGN #1 TO "ktrue:F8,1"
430     DISP "Reading rate gyro history....Please wait."
440     READ #1;W1(*),W2(*),W3(*)
450     ASSIGN #1 TO *
460     Wread=1
470     DISP
480     GOTO Star_cat
490     !
500 Plain_rates:      !
510     Wread=0
520     MAT W1=ZER
```

```

530 MAT W2=(2*PI/5400)      ! One rotation every 90 minutes.
540 MAT W3=ZER
550 !
560 Star_cat: !
570 PRINT USING "/K/K";"Place star catalog disk (Filenames: 'Tab22' and 'Miss2
20') in :F8,1...", "...then push CONT."
580 PAUSE
590 !
600 N$="N"
610 INPUT 'Has table of star catalog cell positions been read-in (Y/N)?',N$
620 PRINT USING "/K,A";"Has Table of star catalog cell positions been read-in?
(Y/N) ";N$
630 !
640 IF N$<>"N" THEN Have_table
650 ASSIGN #1 TO "Tab22:F8,1"
660 READ #1,1
670 READ #1;Table(*)      ! Read in table of cell positions.
680 ASSIGN #1 TO *        ! Close this file.
690 !
700 Have_table:          ! Come here if table has been read-in.
710 ASSIGN #1 TO "Miss220:F8,1"
720 !
730 Num=30              ! Number of records.
740 Len=2048            ! Bytes/record.
750 Fn$="BCdata:F8,1"   ! Dummy file name.
760 INPUT 'File name for simulation run ('Simnnn:F8,1'...where nnn is 3 num.):
",Fn$
770 ON ERROR GOTO Error
780 PRINT USING "/K/K";"File name for simulation run ('Simnnn:F8,1'...where nn
n is 3 num.):",Fn$
790 !
800 Done: !
810 ASSIGN #2 TO Fr$
820 CHECK READ #2
830 OFF ERROR
840 GOTO Ok
850 !
860 Error: !
870 PRINT USING "/K";ERRM$
880 IF ERRN<>55 THEN STOP
890 INPUT 'Should I create this file for you?',N$
900 N$="N"
910 PRINT USING "/K,K";"Should I create this file for you? ",N$
920 IF N$='N' THEN STOP
930 CREATE Fn$,Num,Len
940 GOTO Done
950 !
960 Ok: !
970 READ #2,1 ! Position pointer at beginning of file.
980 File=1
990 !
1000 Continue: !
1010 ! Set satellite orbit parameters.
1020 ! See write-up on "Orbit" for explanation.
1030 As=6652.56      ! Satellite major axis.

```

```

1040  Us=631.3487
1050  Us=Us*Us      ! G*M.
1060  Incl=7.5      ! Incl is satellite inclination in degrees.
1070  !
1080  Ps0(1)=Rs
1090  Ps0(2)=0      ! Initial position of satellite (km).
1100  Ps0(3)=0
1110  !
1120  Vs0(1)=0
1130  Vs0(2)=7.7406*COS(Incl*PI/180) ! Set initial satellite velocity (km/sec).
1140  Vs0(3)=7.7406*SIN(Incl*PI/180)
1150  !
1160  Rs0=SQR(DOT(Ps0,Ps0))
1170  Vsi=SQR(DOT(Vs0,Vs0))      ! Constants needed by Orbit.
1180  Ds0=DOT(Ps0,Vs0)
1190  As=1/(2/Rs0-Vsi*Vsi/Us)
1200  Period=SQR(4*PI*As/Ls)*As*2*PI
1210  PRINT USING Form4;"Period :",Period/60,"minutes"
1220  !
1230  !      Set parameters for earth orbit.
1240  Ae=1.495973710E8      ! Earth orbit major axis (km).
1250  Ue=3.642972685E5
1260  Ue=Ue*Ue
1270  Ve0=29.7845984E      ! Total earth velocity (km/sec).
1280  Earth_incl=23.5
1290  !
1300  Pe0(1)=-Ae      ! Put earth at vernal equinox for initial
1310  Pe0(2)=0      ! position (km).
1320  Pe0(3)=0
1330  !
1340  Ve0(1)=0      ! Initial earth velocity (km/sec).
1350  Ve0(2)=Ve0*COS(-Earth_incl*PI/180) ! Earth is heading downward in
1360  Ve0(3)=Ve0*SIN(-Earth_incl*PI/180) ! inertial frame.
1370  !
1380  Re0=SQR(DOT(Pe0,Pe0))
1390  Vei=SQR(DOT(Ve0,Ve0))      ! Constants needed by Orbit.
1400  De0=DOT(Pe0,Ve0)
1410  Ae=1/(2/Re0-Vei*Vei/Ue)
1420  !
1430  C=3E5      ! C=Velocity of light (km/sec).
1440  Del=4.848E-6 ! = 1 arcsecond in radians.
1450  Omega=2*PI/Period ! Angular frequency of satellite (rad/sec).
1460  !
1470  !      Compute various Euler parameter values.
1480  !
1490  !      Euler parameters relating vehicle and gyro frames.
1500  MAT Bugnom=ZER
1510  Bugnom(1)=1      ! Nominal values.
1520  !
1530  MAT Eug=ZER
1540  N$="N"
1550  INPUT "Do you want variations in Euler para. relating V frame to Gyro fram
e (Y/N)?",N$
1560  PRINT USING " /K,A";"Do you want variations in Euler para. relating V frame
to Gyro frame (Y/N)?",N$

```

```

1570 IF N$='N' THEN No_vg
1580 Evg(1)=Del
1590 Evg(2)=-2*Del      ! Amplitude of variations (radians).
1600 Evg(3)=3*Del
1610 Evg(4)=4*Del
1620 !
1630 Nvg(1)=3
1640 Nvg(2)=4      ! Frequency of variations.
1650 Nvg(3)=5
1660 Nvg(4)=6
1670 !
1680 No_vg:
1690 ! Compute the orientation of FOV(B) w.r.t. FOV(A).
1700 !
1710 Bbanom(1)=1/SQR(2) ! SQR(1/SQR(2)+1)/2
1720 Bbanom(2)=1/SQR(2) ! Bbanom(1) ! Nominal values.
1730 Bbanom(3)=3 ! 1/(SQR(2)*4*Bbanom(1))
1740 Bbanom(4)=3 ! Bbanom(3)
1750 !
1760 MAT Eba=ZER
1770 N$="N"
1780 INPUT 'Do you want variations in Euler para. relating B frame to A frame (Y/N)?',N$
1790 PRINT USING "/K,K";"Do you want variations in Euler para. relating I frame to A frame (Y/N)?",N$
1800 IF N$='N' THEN No_ba
1810 Eba(1)=10*Del
1820 Eba(2)=-30*Del      ! Amplitude of variations (radians).
1830 Eba(3)=20*Del
1840 Eba(4)=5*Del
1850 !
1860 Nba(1)=2
1870 Nba(2)=3      ! Frequency of variations (Oscillations/orbit).
1880 Nba(3)=5
1890 Nba(4)=4
1900 !
1910 No_ba:
1920 ! Set gyro bias values.
1930 MAT Biasnom=ZER
1940 MAT Ebias=ZER
1950 Bias$='N'
1960 INPUT 'Do you want time varying gyro biases (Y/N)?',Bias$
1970 PRINT USING "/K,K";"Do you want time varying gyro biases (Y/N)?",Bias$
1980 IF Bias$='Y' THEN No_bias
1990 !
2000 Biasnom(1)=-Del
2010 Biasnom(2)=2*Del ! Nominal values for biases (radians/sec).
2020 Biasnom(3)=-3*Del
2030 !
2040 Ebias(1)=Biasnom(1)/2
2050 Ebias(2)=-Biasnom(2)/3 ! Amplitude of variations (radians/sec).
2060 Ebias(3)=Biasnom(3)/4
2070 !
2080 Nbias(1)=4
2090 Nbias(2)=6      ! Frequency of variations (Oscillations/orbit).

```

```

2100 Nbias(3)=0
2110
2120 No_bias:      ! Set various constants.
2130
2140 Sigxy=3.4E-3   ! Standard deviation of image coordinates in mm.
2150 Sigm=.05       ! Sigm is the deviation in magnitude.
2160
2170 Xynoise$="Y"
2180 INPUT 'Do you want to add noise to image coordinates (Y/N)?',Xynoise$
2190 PRINT USING "/K,A";"Do you want to add noise to image coordinates(Y/N)?",X
    ynoise$
2200 IF Xynoise$="N" THEN Sigxy=0
2210
2220 F1=72.425      ! Focal length of star sensor lens (mm).
2230 Sigb=PI/180    ! Standard deviation in Euler parameters. This is used
2240                ! if Process B is to run separately.
2250
2260 Siggy=1        ! Standard deviation of gyro noise (degrees/hour).
2270 Siggy=Siggy*PI/(180*3600) ! (rad./second)
2280
2290 Gyro$="N"
2300 INPUT 'Do you want noise added to rate gyro data(Y/N)?',Gyro$
2310 PRINT USING "/K,A";"Do you want noise added to rate gyro data (Y/N)?",Gyro
    $
2320 IF Gyro$="Y" THEN Siggy=0
2330
2340 Sigma=0
2350 Radius=6*PI/180 ! Angle from FOV center for accepting catalog stars.
2360
2370 PLOTTER IS 'GRPHICS'
2380 LORG 5
2390
2400                ! Get the initial gyro orientation w.r.t. the inertial system.
2410
2420 T0=0           ! Set reference time.
2430 Time=0        ! Set initial time (seconds) for this run.
2440
2450 CALL Orbit(Time,Ps(*),Vs(*),Ps0(*),Vs0(*),T0,Us,As,Ds0,Rs0)
2460 PRINT USING Form1;"Position (km) and velocity (km/sec) of Satellite:",Ps(*
    ),Vs(*)
2470
2480 Sc=SQR(DOT(Ps,Fs)) ! Normalize the satellite position.
2490 MAT G3=Ps/(Sc)      ! G3 is along position vector.
2500 CALL Cross(G2(*),G3(*),Vs(*)) ! G2 is normal to orbit plane defined
2510 G2=SQR(DOT(G2,G2)) ! by position and velocity vectors.
2520 MAT G2=G2/(G2)      ! Normalize G2.
2530 CALL Cross(G1(*),G2(*),G3(*)) ! G1 = G2 cross G3.
2540
2550 FOR I=1 TO 3        ! Fill the Gn(*) rotation matrix.
2560     Gn(1,I)=J1(I)
2570     Gn(2,I)=J2(I)
2580     Gn(3,I)=J3(I)
2590 NEXT I
2600
2610 PRINT USING Form2;"Matrix GN:",Gn(*)

```

```

2620      !
2630      ! Compute Euler parameters for VN rotation.
2640      ! First get true Euler parameters for VG rotation.
2650      !
2660      CALL Perturb(Bugnom(*),Evg(*),Nvg(*),Omega,0,Bug(*))
2670      PRINT USING Form3;"Bug...nominal Euler Parameters between frames V-C:",Bug
nom(*)
2680      PRINT USING Form3;"Bug...true Euler Parameters between frames V-G:",Bug(*)
2690      !
2700      CALL Dircosb(Bug(*),Vg(*))
2710      !
2720      MAT Vn=Vg*Gr
2730      ! Recover Euler parameters.
2740      Bun(1)=.5*30R(Vn(1,1)+Vn(2,2)+Vn(3,3)+1)
2750      B0=Bun(1)
2760      Bun(2)=(Vn(2,3)-Vn(3,2))/(4*B0)
2770      Bun(3)=(Vn(3,1)-Vn(1,3))/(4*B0)
2780      Bun(4)=(Vn(1,2)-Vn(2,1))/(4*B0)
2790      !
2800      PRINT USING Form3;"Bun...Initial Euler Parameters between frames V-t:",Bun
(*)
2810      !
2820      !
2830      Step=30      ! seconds between frames.
2840      Delt=.5      ! seconds between gyro readouts.
2850      !
2860      ! Save all constants for this run.
2870      PRINT #2,1;Hs,Us,Ps0(*),Vs0(*),Re,Ue,Pe0(*),Ve0(*),Incl,Sigxy,Sigm,Sigb,Si
ggy,Delt,Time,Step
2880      PRINT #2;Boanon(*),Eba(*),Nba(*),Biasnom(*),Ebias(*),Nbias(*),Bugnom(*),Ev
g(*),Nvg(*)
2890      !
2900      ! Now generate frames of data.
2910      !
2920      TimeLoop:
2930      Ii=0
2940      FOR It=2 TO 30      ! number of frames.
2950      !
2960      DISP ' FRAME: ";It
2970      !
2980      PRINT USING "/K,DD,K";"***** FRAME :";It," *****"
2990      ! Loop over time interval to get true gyro rates.
3000      FOR J=1 TO Step/Delt+1
3010      R1(J)=W1(Ii+J)
3020      R2(J)=W2(Ii+J)
3030      R3(J)=W3(Ii+J)
3040      NEXT J
3050      !
3060      Ii=Ii+Step/Delt
3070      Dt=Time-T0
3080      ! Compute the VG Euler parameters.
3090      CALL Perturb(Evgnom(*),Evg(*),Nvg(*),Omega,Dt,Bug(*))
3100      CALL Dircosb(Evg(*),Vg(*))
3110      PRINT USING Form2;"Matrix VG:",Vg(*)
3120      ! Call Runge-Kutta to compute true values vehicle frame. (We

```

```

3130      ! keep VG fixed during this time interval.)
3140 CALL Runge(Bur(*),Time,Delt,Step,R1(*),R2(*),R3(*),Vg(*))
3150      !
3160 PRINT USING Fcrr4;"Satellite time from start of simulation: ",Time,"seconds"
3170 PRINT USING Fcrr3;"Bun...True Euler Parameters between frames V-N:",Bun(*)
3180      !
3190 CALL Dircosb(Eun(*),Vn(*))
3200      ! Perturb Beta(BA) about their nominal values.
3210 CALL Perturb(Ibanom(*),Eba(*),Nba(*),Omega,Dt,Bba(*))
3220 CALL Dircosb(Eba(*),Ba(*))      ! Compute rotation matrix.
3230      !
3240 PRINT USING Fcrr3;"Bba...Nominal Euler Parameters between frames B-A:",Bbanom(*)
3250 PRINT USING Fcrr3;"Bba...True Euler Parameters between frames B-A:",Bba(*)
3260      !
3270 CALL Dircosb(Eba(*),Ba(*))
3280      !
3290 PRINT USING Fcrr2;"Matrix BA :",Ba(*)
3300      !
3310 CALL Mat_av(Ba(*),Av(*))
3320      ! Compute AN rotation matrix.
3330 MAT An=Av*Vn
3340      ! Update the satellite and earth position.
3350 CALL Orbit(Time,Ps(*),Vs(*),Ps0(*),Vs0(*),T0,Us,As,Ds0,Rs0)
3360 CALL Orbit(Time,Pe(*),Ve(*),Pe0(*),Ve0(*),T0,Ue,Ae,De0,Re0)
3370      !
3380 PRINT USING Fcrr1;"Position (km) and velocity (km/sec) of Satellite:",Ps(*),Vs(*)
3390 PRINT USING Fcrr1;"Position (km) and velocity (km/sec) of Earth:",Pe(*),Ve(*)
3400      !
3410 MAT V=Vs+Ve      ! Compute total velocity.
3420      !
3430 PRINT USING Fcrr1;"Total velocity of satellite (km/sec):",V(*)
3440      !
3450 Mag=SQR(DOT(V,V))
3460 MAT Voc=V/(Mag)
3470      !
3480 FOR I=1 TO 4      ! Perturb Euler parameters.
3490     CALL Gauss(Sigb,Bun(I),Bunest(I))      ! These are used if only Proc. B
3500 NEXT I      ! is run.
3510      !
3520 Mag=SQR(DOT(Bunest,Bunest))      ! Normalize Euler parameters.
3530 MAT Bunest=Bunest/(Mag)
3540      !
3550 File=File+1
3560 PRINT #2,File      ! Position file pointer.
3570 PRINT #2;Bun(*)      ! Save true Euler parameters.
3580 PRINT #2;Bunest(*)      ! Save estimated Euler parameters.
3590 PRINT #2;Bvg(*)      ! Save Betas for frame G to V: Ivg.
3600 PRINT #2;Bvg(*)      ! Save estimated Bvg.
3610 PRINT #2;Bba(*)      ! Save Betas for frame A to B - Bba

```

```

3620 PRINT #2;Bba(*)           ! Save Bba(est.).
3630 PRINT #2;V(*)             ! Save velocity vector.
3640
3650 GCLEAR
3660 LOCATE 10,110,50,100
3670
3680 Pass=0
3690 Pass_2:                   ! Come here for FOV(B)--Second pass through loop.
3700
3710 SHOW -5.7,5.7,-4.4,4.4
3720 CLIP -5.7,5.7,-4.4,4.4
3730 FRAME
3740 AXES 1,1,0,0
3750
3760 FOR J=1 TO 3
3770   Bore(J)=An(3,J)         ! Boresight direction cosines.
3780 NEXT J
3790
3800 PRINT USING Form2;"Matrix AN:",An(*)
3810 PRINT USING Form2;"Boresight unit vector:",Bore(*)
3820
3830 CALL Access(#1,Fov(*),Nfov,Bore(*),Sigma,Radius,Fld,Table(*),Voc(*))
3840
3850 Nm=0
3860 PRINT USING Form4;"Number of stars from the catalog:",Nfov
3870 MAT Xyt=ZER
3880 MAT Xym=ZER
3890
3900 FOR N=1 TO Nfov           ! Find stars that are within FOV.
3910   FOR J=1 TO 3
3920     L(J)=Fov(N,J)
3930   NEXT J
3940   CALL Phozqn(L(*),An(*),Xx,Yy) ! Compute x,y.
3950   Xx=Xx*F1
3960   Yy=Yy*F1
3970   IF ABS(Xx)>5.7 THEN Skip      ! Test for star in field.
3980   IF ABS(Yy)>4.4 THEN Skip
3990
4000   Nm=Nm+1                    ! Got one!
4010   Xyt(Nm,1)=Xx
4020   Xyt(Nm,2)=Yy
4030   CALL Gauss(Sigxy,0,X)      ! Apply noise to positions.
4040   Xym(Nm,1)=Xx+X
4050   CALL Gauss(Sigxy,0,Y)
4060   Xym(Nm,2)=Yy+Y
4070   Mt=Fov(N,4)                ! Save the magnitude.
4080   Xyt(Nm,3)=Mt
4090   CALL Gauss(Sigm,0,Dm)      ! Add noise to magnitude.
4100   Xym(Nm,3)=Mt+Dm
4110
4120   MOVE Xym(Nm,1),Xym(Nm,2)
4130   LABEL USING "K";"+"       ! Plot the star position.
4140   IF Nm=10 THEN Output
4150 Skip:
4160 NEXT N

```

```

4170      !
4180 Output: !
4190 PRINT USING Form4;"Number of stars in this FOV:",Nm
4200 PRINT USING Form2;"True image coordinates (mm):",Xyt(*)
4210 PRINT USING Form2;"Measured image coordinates (mm):",Xym(*)
4220      !
4230 PRINT #2;Nm                      ! Save number of stars in field.
4240 PRINT #2;Xyt(*),Xym(*)          ! Save true and measured positions.
4250      !
4260 MAT Bn:=Ba*An                    ! Rotate to FOV(B).
4270 MAT An:=Bn
4280 LOCATE 10,110,0,50
4290      !
4300 Pass=Pass+1
4310 IF Pass=1 THEN Na=Nm
4320 IF Pass=1 THEN Pass_2
4330      !
4340 IF Gyro$="N" THEN Skip_noise
4350 DISP "Adding noise to gyro rates...please wait."
4360      !
4370 FOR J=1 TO Step/Delt
4380     CALL Gauss(Siggy,0,X)
4390     R1(J)=R1(J)+X
4400     CALL Gauss(Siggy,0,X)
4410     R2(J)=R2(J)+X
4420     CALL Gauss(Siggy,0,X)
4430     R3(J)=R3(J)+X
4440 NEXT J
4450      !
4460 Skip_noise: !
4470 IF Bias$="N" THEN Skip_bias
4480      ! Calculate the bias rates.
4490     CALL Bias(Dt,Biasnom(*),Ebias(*),Nbias(*),Omega,Bias(*))
4500     PRINT USING Form3;"Biases...true values:",Bias(*)
4510     MAT R1=R1+(Bias(1))
4520     MAT R2=R2+(Bias(2))
4530     MAT R3=R3+(Bias(3))
4540 Skip_bias: !
4550      !
4560 OUTPUT 0 USING "K,DD,K,DD,XX,DD";"Frame: ",It," Number of stars: ",Na,Nm
4570 DISP
4580      !
4590 PRINT #2;R1(*),R2(*),R3(*)        ! Save measured gyro rates (60*3)
4600 PRINT #2;Bias(*)                 ! Save true bias values (3).
4610      !
4620 NEXT I:
4630      !
4640 ASSIGN * TO #2
4650      !
4660 N$="N"
4670 INPUT "Do you want to do another run(Y/N)?",N$
4680 PRINT USING "K,K";"Do you want to do another run(Y/N)? ";N$
4690 IF N$="Y" THEN Restart
4700      !
4710 Form1: IMAGE K/3(3(MD.DDDDE,X)/)

```

4720 Form2: IMAGE K/10<3<MD.DDDDDD,X>/>
4730 Form3: IMAGE K/<4<MD.DDDDDD,X>/>
4740 Form4: IMAGE K,X,MDDDD.DD,X,K
4750 |
4760 END

Combin

This program analyses data produced by simulation program Datgen, directing the data to subroutines for Process B (Proc-b) and Process C (Orbit, Runge and Kalman). Process B and Process C can be run separately (Process C alone only if Process B was run previously with these data) or they can be run together. Combin requests several parameters from the user: 1) the process noise standard deviation associated with the variation in interlock Euler parameters, β_{BA} , used in Kalman filter update of β_{BA} (see Section 3 of the Final Report), and 2) the gyro bias "standard deviation" which controls the variations in the recovered gyro biases (see Section 4 of the Final Report). In addition, we can also offset, by a constant amount, the interlocks between the vehicle and gyro frame (nominally set to zero).

The data frames, read from an external file, are processed one at a time. For each frame, the Euler parameters, β_{VN} , and the associated covariance matrix from analysis of the previous frame, are integrated forward by subroutine Runge (for the first frame we can use the true values for β_{VG} or some offset). Subroutine Orbit computes the position and velocity of the earth and satellite (the total velocity is used by Access to add aberration effects to the star direction cosines). We then call Proc-b to 1) match measured stars with specific catalog stars and 2) update the Euler parameters β_{VN} and β_{BA} via least-squares correction. These Euler parameters, the covariance matrix associated with β_{VN} , and the calculated image coordinates for all matched stars are saved by storing them at the end of the current data record.

Subroutine Kalman is then called to combine the integrated values for β_{VN} with the corrected values from Process B analysis to yield an "optimal estimate" of β_{VN} and the gyro biases, at the current time. These parameters and the 4×4 covariance matrix associated with the estimate of β_{VN} are saved on the data file also.

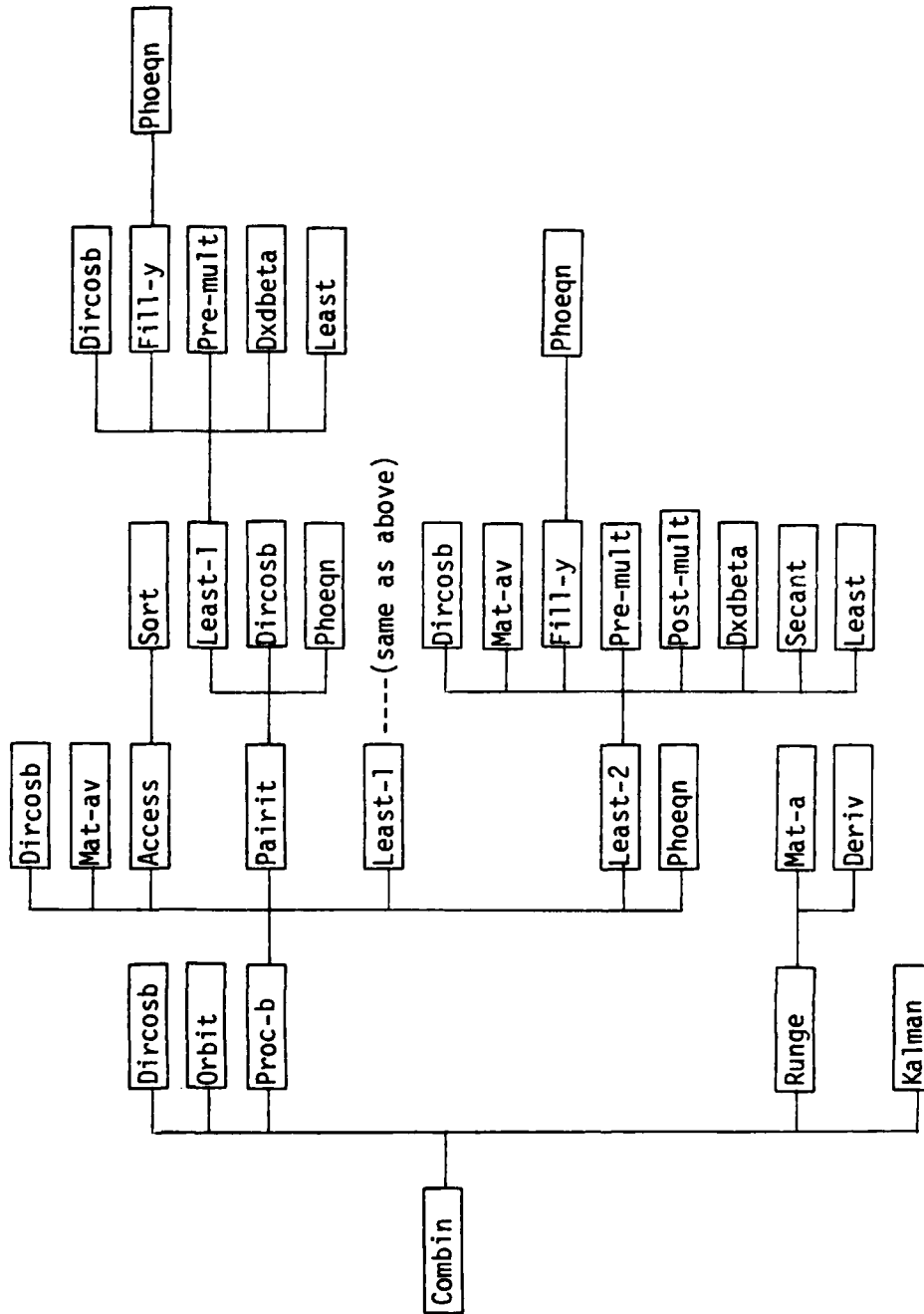


Figure A10.2: Hierarchy of subroutine calls (left to right) and approximate order of calls top to bottom) for program Combin.

```

*****
*
*           P R O G R A M : C O M B I N E
*
*****

80      !   T. STRIKWERDA ..... 14 JANUARY 1981.
90      !   This program combines processes B and C of Star Wars. This versior uses
100     !   Euler parameters relating the "V" frame to the inertial frame.
110     !   This program also recovers the Euler parameters relating FOV(A) to
120     !   FOV(B) and the gyro bias estimates.
130     !
140     OVERLAP
150     FLOAT 15
160     OPTION BASE 1
170     DIM Ps0(3),Vs0(3),Ps(3),Vs(3),Pe0(3),Ve0(3),Pe(3),Ve(3)
180     DIM T1(4),T2(3,3)
190     DIM Cov8(8,8)
200     DIM Bun(4),Buntrue(4),Bug(4),Buns(4)
210     DIM Bba(4),Bbatrue(4),Voc(3),Bbae(4),Bbalsq(4)
220     SHORT Xyma(10,3),Xymb(10,3),Xyca(5,2),Xycb(5,2),Xyt(10,3)
230     SHORT Cov(4,4),W1(61),W2(61),W3(61),Short
240     DIM Pk(7,7),Lan(7,7)
250     DIM Xks(7),Xki(7),Xkb(7),Q(3,3),Qba(4,4),Pba(4,4)
260     COM Vg(3,3),INTEGER Table(529,2)
270     RAD
280     !
290     PRINT USING "K";"***** P R O C E S S   B   A N D   C
*****"
300     !
310     N$="N"
320     INPUT "Doing Proc B (Y/N)?",N$
330     PRINT USING "/K,A";"Doing Proc B (Y/N)? ",N$
340     Pb=0
350     IF N$="Y" THEN Pb=1
360     !
370     N$="N"
380     INPUT "Doing Proc C (Y/N)?",N$
390     PRINT USING "/K,A";"Doing Proc C (Y/N)? ",N$
400     Pc=0
410     IF N$="Y" THEN Pc=1
420     !
430     IF Pb=0 THEN Have_table
440     !
450     PRINT USING "K";"Insert star catalog disk into F8,1....Then press CONT"
460     PAUSE
470     N$=" " ! Set answer to blank.
480     INPUT "Has the table of star cell positions been read-in?",N$
490     PRINT USING "/K,K";"Has cell table been read-in? ",N$
500     IF N$<>"Y" THEN Have_table
510     !
520     ASSIGN #1 TO "Tab22:F8,1"
530     READ #1,1

```

```

540 READ #1;Table(*)      ! Read in table of cell positions.
550 ASSIGN * TO #1        ! Close this file.
560 !
570 ASSIGN #2 TO "Mss220:F8,1" ! Star catalog file.
580 !
590 Have_table: ! Come here if this is a continuation.
600 INPUT "Input file name and device with simulation data (Simnnn:F8,0):",D
in$
610 PRINT USING "/K,K";"Input file name and device with simulation data: ",D
in$
620 ASSIGN #1 TO Din$
630 ! BUFFER #1
640 CHECK READ #1
650 !
660 READ #1,1;As,Us,Ps0(*),Vs0(*),Ae,Ue,Pe0(*),Ve0(*),Incl,Sigxy,Sigm,Sigb,S
iggy,Delt,Time,Step
670 PRINT USING "/K,DDDD";"Satellite orbit major axis (km): ",As
680 PRINT USING "K,D.DDDE";"Earth orbit major axis (km): ",Ae
690 PRINT USING "K,DDDD";"Satellite orbit inclination (deg.):",Incl
700 Sigxy=MAX(Sigxy,1E-3)
710 PRINT USING "K,DD.DD";"Rate gyro data spacing (sec): ",Delt
720 PRINT USING "K,DD.DD";"Runge-Kutta time step (sec): ",Step
730 PRINT USING "K,D.DDDE";"Gyro standard deviation (rad/sec): ",Siggy
740 !
750 Siggy=MAX(Siggy,2.424E-6) ! Must keep gyro std. dev. non-zero.
760 !
770 Ds0=DOT(Ps0,Vs0)
780 Rs0=SQR(DOT(Fs0,Ps0))      ! Constants for orbit calculation.
790 De0=DOT(Pe0,Ve0)
800 Re0=SQR(DOT(Fe0,Pe0))
810 T0=0                      ! Set reference time.
820 !
830 REDIM W1(33)
840 READ #1;41(*)      ! Read all the perturbation constants.
850 !
860 FOR I=1 TO 4
870     Bbarom(I)=W1(I)      ! Retrieve the nominal interlock values
880 NEXT I                  ! from this list.
890 !
900 REDIM W1(61)
910 !
920 IF Pb=0 THEN READ #1;Wba      ! Read the weight for interlock recovery.
930 IF Pb=0 THEN Skip1
940 !
950 INPUT "Input weight in arcseconds for interlock variance (2,5,etc.)",Wba
960 PRINT USING "/K,DD.DDD";"Input weight in arcseconds for interlock varian
ce (2,5,etc.) ",4ba
970 PRINT #1;Wba
980 MAT Pba=IIN              ! Set up interlock Kalman filter matrices.
990 MAT Pba=Pba*(1E-6) ! Set covariance matrix to large initial value.
1000 !
1010 MAT Qba=IIN
1020 Qba(1,1)=1/8
1030 Qba(1,2)=-1/8
1040 Qba(2,1)=-1/8          ! Qba is the process noise matrix for the

```

```

1050 Qba(2,2)=1/8      ! interlock angles between FOV(A) and FOV(B).
1060 Qba(3,3)=1/4      ! The Q matrix has been converted to Euler
1070 Qba(4,4)=1/4      ! parameters. NOTE: This matrix is valid for
1080 MAT Qba=Qba*((Wba*4.848E-6)^2) ! (3,1,3) rotation of (0,90,0) only!
1090 MAT Pba=Qba      ! Can start with good estimate.
1100 !
1110 Skip1: !
1120 W=1E1      ! SQR(Weight) for constraint equation.
1130 !
1140 IF P=0 THEN Skip2
1150 !
1160 INPUT "Input Gyro Bias Standard Deviation (Degrees/Hr)",Sigbias
1170 PRINT USING "/K,DD.DD";"Input Gyro Bias Standard Deviation (Degrees/Hr
) ",Sigbias
1180 PRINT #1;Sigbias
1190 Sigbias=Sigbias*PI/(180*3600)      ! Convert to radians/sec.
1200 !
1210 MAT Lam=ZER
1220 FOR I=5 TO 7
1230   Lam(I,I)=Sigbias^2      ! Set observation covariance matrix for gyro bias.
1240 NEXT I
1250 !
1260 MAT Pk=IDN
1270 MAT Pk=P+(1E-7)      ! Set covariance to large value.
1280 !
1290 Qfac=Siggy^2
1300 MAT Q=IDN
1310 MAT Q=Q*(Qfac)      ! Initialize process noise matrix.
1320 PRINT USING Form3;"Q Matrix:",Q(*)
1330 !
1340 MAT Bug=ZER
1350 Bug(1)=1
1360 N$="N"
1370 INPUT "Do you want to offset matrix VG (V-frame to Gyro frame)",N$
1380 PRINT USING "/K,K";"Do you want to offset matrix VG (V-frame to Gyro fra
me) ";N$
1390 !
1400 IF N$="N" THEN GOTO Skipvg
1410 Bug(1)=1
1420 Bug(2)=1E-3      ! Size of offset can be changed.
1430 Bug(3)=-2E-3
1440 Bug(4)=3E-3
1450 Mag=SQR(DOT(Bug,Bug))      ! Re-normalize Euler parameters.
1460 MAT Bug=Bug/(Mag)
1470 Skipvg: !
1480 CALL Dir:osb(Bug(*),Vg(*),0,T2(*),T2(*),T2(*),T2(*))
1490 PRINT USING Form3;"Matrix VG for this run:",Vg(*)
1500 !
1510 Skip2: !
1520 !
1530 C=3E5      ! Speed of light (km/sec).
1540 Pass=0      ! First pass through frame loop - special case.
1550 Tk=Time      ! Initial time.
1560 MAT Kki=ZER
1570 MAT Kkt=ZER

```

```

1580      !
1590      ! ++++++ Begin Loop Over Data Frames. ++++++
1600      !
1610      FOR Ifile=2 TO 30      ! Loop over all data frames.
1620          PRINT USING "/K,DDD,K";"~~~~~ RECORD NUMBER: ",Ifile," ~~~~~"
1630          Pass=Pass+1
1640          !
1650          READ #1,Ifile
1660          READ #1;Buntrue(*)
1670          PRINT USING Form1;"Bun....True Euler parameters between V and N frames:"
, Buntrue(*)
1680          REDIM Xkt(4)
1690          MAT Xkt=Buntrue
1700          REDIM Xkt(7)
1710          IF Pass=1 THEN MAT Xki=Xkt ! This causes displacement of first estimate.
1720          !
1730          READ #1;Bun(*)      ! These are not used unless we do Proc. B only.
1740      ! PRINT USING Form1;"Bun....Estimated Euler parameters between V and N fra
mes:",Bun(*)
1750          !
1760          READ #1;Bug(*)      ! True Euler parameters between V and G frames.
1770          ! CALL DirCosb(Bug(*),Vg(*),0,T2(*),T2(*),T2(*),T2(*)) ! Could use truth
.
1780          READ #1;Bug(*)      ! same comment as Bun....not used.
1790          !
1800          READ #1;Bbatrue(*) ! True Euler parameters between B and A frame.
1810          READ #1;Bbatrue(*) ! same....not used.
1820          IF Pass=1 THEN MAT Bba=Bbatrue ! Can help out by setting estimate=truth.
1830          !
1840          PRINT USING Form1;"Bba....True Euler parameters between B and A frames:"
,Bbatrue(*)
1850          PRINT USING Form1;"Bba....Current Euler parameters between B and F frame
s:",Bba(*)
1860          !
1870          READ #1;Voc(*)
1880          PRINT USING Form3;"Components of total velocity (km/sec):";Voc(*)
1890          MAT Voc=Voc/(C)
1900          !
1910          REDIM Xyna(1E1,3),Xymb(1E1,3)
1920          PRINT USING "K";"Number of stars in each FOV:"
1930          !
1940          READ #1;Nfovna
1950          PRINT USING "K,X,DD";" FOV(A):",Nfovna
1960          READ #1;Xyna(*)      ! True coordinates.
1970          READ #1;Xyma(*)      ! Measured coordinates.
1980          !
1990          READ #1;Nfoymb
2000          PRINT USING "K,X,DD";" FOV(B):",Nfoymb
2010          READ #1;Xymb(*)      ! True.
2020          READ #1;Xymb(*)      ! Measured.
2030          !
2040          READ #1;W1(*),W2(*),W3(*) ! Read rate gyro data for each axis.
2050          !
2060          REDIM Xkt(7)
2070          READ #1;Xkt(5),Xkt(6),Xkt(7) ! Read true bias rates.

```

```

2080      !
2090      IF Pc=0 THEN Skip3
2100      CALL Runge(Tk,Delt,Step,W1(*),W2(*),W3(*),Xki(*),Pk(*),Q(*),Sigbias)
2110      !
2120      PRINT USING Form2;"Bun....Integrated Euler parameters between V and N fr
ames","and gyro biases:",Xki(*)
2130      !
2140      REDIM Xki(4)
2150      MAT Bun=Xki      ! Estimates for Process B.
2160      ! IF Pass=1 THEN MAT Bun=Buntrue ! Can help out by setting est.=truth.
2170      REDIM Xki(7)
2180      !
2190      Skip3: IF Pc=0 THEN Tk=Tk+Delt
2200      CALL Orbit(Tk,Ps(*),Vs(*),Ps0(*),Vs0(*),T0,Us,As,Ds0,Rs0)
2210      CALL Orbit(Tk,Pe(*),Ve(*),Pe0(*),Ve0(*),T0,Ue,Ae,De0,Re0)
2220      MAT Voc=Vs+Ve      ! Total velocity.
2230      PRINT USING Form3;"Components of total velocity (km/sec):";Voc(*)
2240      MAT Voc=Voc/C
2250      !
2260      IF Pb=0 THEN Skip4
2270      !
2280      MAT Bbae=Bba      ! Save the estimated interlock vector.
2290      CALL Proc_b(#2,Bun(*),Bba(*),Voc(*),Nfovma,Nfovmb,Ka,Kb,W,Sigxy,Xyma(*),
Xymb(*),Xyca(*),Xycb(*),Cov8(*),Pba(*),Qba(*),Bbalsq(*))
2300      ! OUTPUT @ USING Form9; Ifile,Nfovma,Nfovmb,Ka,Kb ! Progress indicator.
2310      Form9: IMAGE 5(DD,XX)
2320      !
2330      FOR I=1 TO 4
2340      FOR J=1 TO 4
2350      Cov(I,J)=Cov8(I,J)
2360      NEXT J
2370      NEXT I
2380      PRINT #1;Bun(*),Bba(*)      ! Save Process B results.
2390      PRINT #1;Cov(*)
2400      PRINT #1;Ka,Xyca(*)
2410      PRINT #1;Kb,Xycb(*)
2420      !
2430      Skip4: IF Pc=0 THEN Endloop
2440      !
2450      IF Pb=0 THEN READ #1;Bun(*),Bba(*),Cov(*),Ka,Xyca(*),Kb,Xycb(*)
2460      IF (Ka=0) AND (Kb=0) THEN Proc_b_failure
2470      REDIM Xkb(4)
2480      MAT Xkb=Bun
2490      REDIM Xkb(7)
2500      !
2510      FOR I=1 TO 4
2520      FOR J=1 TO 4
2530      Lam(I,J)=Cov(I,J)      ! Fill obs. cov. matrix with Proc. B results.
2540      NEXT J
2550      NEXT I
2560      !
2570      REDIM Xki(4)
2580      MAT T1=Xki
2590      REDIM Xki(7)
2600      !

```

```

2610 CALL Kalnan(Xkt(*),Xki(*),Pk(*),Xkb(*),Lam(*))
2620 !
2630 Proc_b_failure: ! Come here if Proc B failed to find stars in both FOV.
2640 !
2650 PRINT #1;Xki(*) ! Save Proc. C results.
2660 PRINT #1;T1(*)
2670 !
2680 FOR I=1 TO 4
2690 FOR J=1 TO 4
2700 Short=Pk(I,J)
2710 PRINT #1;Short
2720 NEXT J
2730 NEXT I
2740 PRINT #1;Bbalsq(*)
2750 !
2760 Endloop: !
2770 PRINT " (end of frame)"
2780 !
2790 NEXT I:file
2800 !
2810 Stop: FRINT
2820 FRINT " THE END "
2830 FOR I=1 TO 5
2840 BEEP
2850 WAIT 120
2860 NEXT I
2870 !
2880 Form1: IMAGE /K,/,4(MD.DDDDDD,XX)
2890 Form2: IMAGE /K,/,K,/,4(MD.DDDDDD,X),/,3(MD.DDDE,X)
2900 Form3: IMAGE /K,3(/3(MD.DDDE,XX))
2910 !
2920 END

```

Access

Access determines which catalog cells the camera boresight lies in or near and then retrieves the star positions for the stars contained in those cells. The first step is to determine the polar angle and longitude angle of the boresight unit vector. These angles are converted to primary cell indices by dividing by the cell size. In a similar manner, we also determine three nearest neighbor cells. The location of each cell in memory or on a storage device and the number of stars in each cell are found by referring to a table. We then read the star data from these cells (consisting of direction cosines and magnitude) and compute the vector dot product of each star with the boresight unit vector. If the product is less than some specified tolerance we reject the star; this product is also used to sort the subcatalog by distance off the boresight. A list of up to 100 stars is returned to the calling program.

See Appendix 3 for details on the catalog format.

```
*****
*
*           A C C E S S
*
*****
```

```
6260 ! Access gets stars from the catalog for cells surrounding the boresight.
6270 SUB Access(I1,Nfov,Bore(*),Sigma,Radius,Voc(*),SHORT Fov(*))
6280 OPTION BASE 1
6290 DIM Ang(100),Cc(3)
6300 INTEGER K(4),J(4),Nt,Kk,M,N
6310 SHORT Mag,Num
6320 COM Vg(3,3),INTEGER Table(529,2)
6330 REDIM Fov(100,4)
6340 !
6350 DISP "Access."
6360 !
6370 Nt=22 ! Number of latitude bands for this catalog.
6380 Dphi=2*PI/(2*Nt+1) ! Latitude spacing.
6390 !
6400 Nfov=0
6410 Ctest=COS(Radius+Sigma) ! Maximum angle off the boresight.
6420 Phi=ACS(Bore(3)) ! Polar angle.
6430 IF Phi<0 THEN Phi=Phi+2*PI
6440 !
6450 Lam=PI/2 ! Calculate longitude angle.
6460 IF Bore(2)<0 THEN Lam=Lam+PI
6470 IF Bore(1)<>0 THEN Lam=ATN(Bore(2)/Bore(1))
6480 IF Bore(1)<0 THEN Lam=Lam+PI
6490 IF Lam<0 THEN Lam=Lam+2*PI
6500 !
6510 PRINT USING "/,K,X,DDD.D,X,K";"Polar angle:",Phi*180/PI,"Degrees"
6520 PRINT USING "K,X,DDD.D,X,K";"Longitude angle:",Lam*180/PI,"Degrees"
6530 !
6540 ! Calculate cell indices.
6550 IF Phi<Dphi THEN North ! Near north pole--special case.
6560 IF Phi>2*PI-Dphi*.5 THEN South ! Near south pole--special case.
6570 !
6580 K(1)=2+INT(Phi/Dphi+.5) ! Calculate two neighboring
6590 K(3)=2+INT(Phi/Dphi) ! latitude bands.
6600 IF K(3)=K(1) THEN K(3)=K(3)+2
6610 IF K(1)>Nt THEN K(1)=2*Nt+1-K(1) ! Make sure we're on the correct
6620 IF K(3)>Nt THEN K(3)=2*Nt+1-K(3) ! side of equator.
6630 K(2)=K(1)
6640 K(4)=K(3)
6650 !
6660 Diam=2*PI/(2*K(1)+1) ! Now determine two neighboring cells
6670 J(1)=INT(Lam/Diam+.5) ! in each latitude band.
6680 J(2)=INT(Lam/Diam)
6690 IF J(2)=J(1) THEN J(2)=(J(1)+1) MOD (2*K(1)+1)
6700 J(1)=J(1) MOD (2*K(1)+1)
6710 Diam=2*PI/(2*K(3)+1)
6720 J(3)=INT(Lam/Diam+.5)
6730 J(4)=INT(Lam/Diam)
6740 IF J(4)=J(3) THEN J(4)=(J(3)+1) MOD (2*K(3)+1)
```

```

6750 J(3)=J(3) MOD (2*K(3)+1)
6760 GOTO Around
6770 !
6780 North: ! Special case for north pole.
6790 K(1)=0
6800 K(2)=2
6810 K(3)=2 ! Get indices for polar cap and
6820 K(4)=2 ! three of the five cells in next band.
6830 J(1)=0
6840 J(3)=INT(Lam/(2*PI/5)+.5)
6850 J(2)=J(3)+4 MOD 5
6860 J(4)=J(3)+1 MOD 5
6870 GOTO Around
6880 !
6890 South: ! Special case for south pole.
6900 K(1)=1
6910 K(2)=1 ! Get all three cells at south pole and
6920 K(3)=1 ! one of the seven in the next band.
6930 K(4)=3
6940 J(1)=0
6950 J(2)=1
6960 J(3)=2
6970 J(4)=INT(Lam/(2*PI/7)+.5)
6980 !
6990 Around: ! Skip north and south pole stuff.
7000 PRINT USING "/,K,/,2(4(DDD,X)/)";"Cell indices:",K(*),J(*)
7010 !
7020 FOR I=1 TO 4
7030 M=K(I)*K(I)+J(I)+1
7040 N=Table(1,1)
7050 READ #1,N
7060 Kk=Table(N,2)
7070 FOR K=1 TO Kk
7080 READ #1;Co(*),Mag,Num
7090 A=DOT(Co,Bcre) ! Compute cos of interstar angle.
7100 IF A<test THEN Continue
7110 Nfov=Nfov+1 ! Star lies within range of boresight.
7120 Ang(Nfov)=F
7130 !
7140 Scale=1-DOT(Voc,Co)
7150 MAT Co=Co*(Scale) ! Add aberration effects.
7160 MAT Co=Co+voc
7170 !
7180 FOR J=1 TO 3
7190 Fov(Nfov,J)=Co(J) ! Save this star.
7200 NEXT J
7210 Fov(Nfov,4)=Mag
7220 Continue: NEXT K
7230 Skip: !
7240 NEXT I
7250 !
7260 IF Nfov<>0 THEN CALL Sort(Ang(*),Fov(*),Nfov,4) ! Sort stars by angle off
boresight.
7270 !
7280 DISP
7290 SUBEND

```

Bias

Subroutine Bias computes bias terms which are later added to the rate gyro data. We have used a simple sinusoidal variation added to each nominal bias value. For input we specify the time, orbital frequency, nominal bias values, and amplitude and frequency of each variation.

```

*****
*
*               B I A S
*
*
*****

```

```

7650 SUB Bias(Dt,SHCRT Biasnom(*),Ebias(*),Nbias(*),REAL Omega,Bias(*))
7660 ! This sub-routine computes the bias rates to be added to gyro rates.
7670 OPTION BASE 1
7680 !
7690 Bias(1)=Biasnom(1)+Ebias(1)*COS(Nbias(1)*Omega*Dt)
7700 Bias(2)=Biasnom(2)+Ebias(2)*SIN(Nbias(2)*Omega*Dt)
7710 Bias(3)=Biasnom(3)+Ebias(3)*SIN(Nbias(3)*Omega*Dt)
7720 !
7730 SUBEND

```

123

Cross

Subroutine Cross computes the vector cross-product of two vectors.

```
*****
*                                     *
*               C R O S S             *
*                                     *
*****
```

```
6290 ! This subprogram computes the cross product of two vectors.
6300 !
6310 SUB Cross(R(*),V1(*),V2(*))
6320 OPTION BASE 1
6330 !
6340 FOR K=1 TO 3
6350     K1=K MOD 3+1 ! Determine the order of multiplication.
6360     K2=K1 MOD 3+1
6370     R(K)=V1(K1)*V2(K2)-V1(K2)*V2(K1)
6380 NEXT K
6390 !
6400 SUBEND
```

Deriv

This subroutine forms the right-hand-side of the matrix Riccati equation, which is integrated by subroutine Runge to propagate the covariance matrix. We use the partitioned form of the Riccati equation as presented in Appendix 8 of the Final Report.

The input data consist of the upper 4×4 portion, lower left 3×4 portion and lower right 3×3 portion of the covariance matrix. In addition, the subroutine requires the 3×3 process noise matrix and two portions of the matrices used for the state differential equations (see Appendix 8). The output is the time derivative of the upper 4×4 and lower left 3×4 portions of the covariance matrix evaluated for the current state.

```

*****
*
*               D E R I V
*
*****

```

```

14890 ! This subroutine sets up the RHS of the matrix Riccati equation in
14900 ! partitioned form.
14910 SUB Deriv(P11(*),P21(*),P22(*),Q(*),A11(*),A12(*),Kn(*),Ln(*))
14920 OPTION BASE 1
14930 DIM T1(4,4),T2(4,4),A12t(3,4)
14940 !
14950 MAT T1=A11*P11
14960 MAT T2=A12*P21
14970 MAT T1=T1+T2
14980 MAT T2=TRN(T1)
14990 MAT Kn=T1+T2
15000 MAT A12t=TRN(A12)
15010 MAT T1=A12*Q
15020 MAT T2=T1+A12t
15030 MAT Kn=Kn+T2
15040 !
15050 MAT T1=P21*A11
15060 MAT T2=P22*A12t
15070 MAT Ln=T2-T1
15080 !
15090 SUBEND

```

Dircosb

This subroutine computes the rotation matrix between two coordinate frames as a function of the Euler parameters. If selected, the partial derivatives of the rotation matrix with respect to each Euler parameters, are calculated also. Refer to Section 2 of the Final Report for the form of the rotation matrix.

```

*****
*
*               I I R C O S B
*
*****

8880  ! Dircosb computes the direction cosine matrix using EULER parameters.
8890  SUB Dircosb(B(*),C(*),Ndc,Dc1(*),Dc2(*),Dc3(*),Dc4(*))
8900  OPTION BASE 1
8910  !
8920  DISP "Dircosb"
8930  !
8940  B0=B(1)
8950  B1=B(2)
8960  B2=B(3)
8970  B3=B(4)
8980  B12=B1+B1
8990  B22=B2+B2
9000  B32=B3+B3
9010  B02=B0+B0
9020  C(1,1)=B02+B12-B22-B32
9030  C(1,2)=2*(B1*B2+B0*B3)
9040  C(1,3)=2*(B1*B3-B0*B2)
9050  C(2,1)=2*(B1*B2-B0*B3)
9060  C(2,2)=B02-B12+B22-B32
9070  C(2,3)=2*(B2*B3+B0*B1)
9080  C(3,1)=2*(B1*B3+B0*B2)
9090  C(3,2)=2*(B2*B3-B0*B1)
9100  C(3,3)=B02-B12-B22+B32
9110  !
9120  DISP
9130  !
9140  IF Ndc<>1 THEN SUBEXIT ! Don't need partials.
9150  !
9160  DISP "Dircosb"
9170  !
9180  B0=B0+B0
9190  B1=B1+B1
9200  B2=B2+B2
9210  B3=B3+B3
9220  !
9230  Dc1(1,1)=B3
9240  Dc1(1,2)=B3
9250  Dc1(1,3)=-B2
9260  Dc1(2,1)=-B3
9270  Dc1(2,2)=B3
9280  Dc1(2,3)=B1
9290  Dc1(3,1)=B2
9300  Dc1(3,2)=-B1
9310  Dc1(3,3)=B3
9320  !
9330  Dc2(1,1)=B1
9340  Dc2(1,2)=B2

```

! Direction cosine matrix.

! Compute the partials of C w.r.t. each beta.

! Do this for factor of two.

! Partial w.r.t. Beta0.

! Partial w.r.t. Beta1.

9350 Dc2(1,3)=B3
 9360 Dc2(2,1)=B2
 9370 Dc2(2,2)=-B1
 9380 Dc2(2,3)=B3
 9390 Dc2(3,1)=B3
 9400 Dc2(3,2)=-B0
 9410 Dc2(3,3)=-B1
 9420 !

! Partial's w.r.t. Beta2.

9430 Dc3(1,1)=-B2
 9440 Dc3(1,2)=B1
 9450 Dc3(1,3)=-B0
 9460 Dc3(2,1)=B1
 9470 Dc3(2,2)=B2
 9480 Dc3(2,3)=B3
 9490 Dc3(3,1)=B3
 9500 Dc3(3,2)=B3
 9510 Dc3(3,3)=-B2
 9520 !

! Partial's w.r.t. Beta3.

9530 Dc4(1,1)=-B3
 9540 Dc4(1,2)=B3
 9550 Dc4(1,3)=B1
 9560 Dc4(2,1)=-B0
 9570 Dc4(2,2)=-B3
 9580 Dc4(2,3)=B2
 9590 Dc4(3,1)=B1
 9600 Dc4(3,2)=B2
 9610 Dc4(3,3)=B3
 9620 !
 9630 DISP
 9640 SUBEND

Dxdbeta

This subroutine computes the partial derivatives of the calculated image coordinates with respect to each of the four Euler parameters orienting the star tracker. We have rearranged the partial derivative calculation into a simple form. We want, for FOV(A) for example,

$$\frac{\partial X_m}{\partial \beta_{VN_n}} = \sum_{ij} \frac{\partial X_m}{\partial AN_{ij}} \frac{\partial AN_{ij}}{\partial \beta_{VN_n}} \quad \begin{array}{l} (m = 1, 2, \dots \text{no. of stars} \\ n = 0, 1, 2, 3). \end{array}$$

We note that matrix $\partial AN / \partial \beta_{VN_n} = AV \cdot \partial VN / \partial \beta_{VN_n}$ where AV is calculated by Mat-av and $\partial VN / \partial \beta_{VN_n}$ is calculated by Dircosb and this matrix is independent of the particular star.

The terms $\partial X_m / \partial AN_{ij}$ are derivatives of the stellar colinearity equations with respect to the terms in the rotation matrix. If we rotate a star's direction cosines {L} into the FOV(A) frame, we have

$$\{L'\} = AN \cdot \{L\}$$

Then, the normalized image coordinate $X/f = L'_1/L'_3$ where f = lens focal length. The 3×3 matrix $\partial X_m / \partial AN$ becomes

$$\begin{bmatrix} \frac{L_1}{L'_3} & \frac{L_2}{L'_3} & \frac{L_3}{L'_3} \\ 0 & 0 & 0 \\ \frac{L_1 L'_1}{L'^2_3} & \frac{L_2 L'_1}{L'^2_3} & \frac{L_3 L'_1}{L'^2_3} \end{bmatrix}$$

or simply the outer product:

$$\begin{Bmatrix} \frac{1}{L'_3} \\ 0 \\ \frac{L'_1}{L'^2_3} \end{Bmatrix} \begin{Bmatrix} L_1 & L_2 & L_3 \end{Bmatrix}$$

To obtain the derivative we can proceed to multiply each term in this 3×3 matrix by the corresponding term in $\partial AN / \partial \beta_{VN_n}$ and then sum all terms. However, it is not too difficult to see that this can be accomplished by writing

$$\frac{\partial X_m}{\partial \beta_{VN_n}} = \left\{ \begin{array}{ccc} 1 & 0 & \frac{L_1^1}{L_3^1} \end{array} \right\} \left[\frac{\partial AN}{\partial \beta_{VN_n}} \right] \left\{ \begin{array}{c} L_1 \\ L_2 \\ L_3 \end{array} \right\} ,$$

a form very suitable for computation. The partial derivatives of the y coordinate are similar.

```

*****
*
*
*
*****
IXDBETA
*****
8350  ! Dxdbeta computes partial derius of (x,y) w.r.t. Euler parameters.
8360  ! d(x or y)/d(Ieta(i))=Sum[(d(x or y)/d(Cij)).(d(Cij)/d(Beta(i)))]
8370  !   where "." represents mult. corresponding terms.
8380  ! We have rearranged this sum into compact form used here--(see
8390  ! notes of T. Strikwerda).
8400  SUB Dxdbeta(Fic(*),A(*),Kk,C(*),Dc1(*),Dc2(*),Dc3(*),Dc4(*))
8410  OPTION BASE 1
8420  DIM T1(3),T3(3),T4(3),L(3)! Dimension temporary matrices.
8430  !
8440  DISP "Dxdbeta"
8450  !
8460  FOR K=1 TO Kk      ! Loop over all stars in this FOV.
8470      K1=K+K-1
8480      K2=K1+1
8490      FOR I=1 TO 3      ! Get direction cosines for this star.
8500          L(I)=Fic(K,I)
8510      NEXT I
8520      MAT T1=C*L        ! Compute direction cosines in FOV frame.
8530      T3(1)=1/T1(3)
8540      T3(2)=0
8550      T3(3)=-T1(1)*T3(1)*T3(1)
8560      T4(1)=0
8570      T4(2)=T3(1)
8580      T4(3)=-T1(2)*T3(1)*T3(1)
8590      !
8600      MAT T1=D:1*L
8610      A(K1,1)=DOT(T1,T3)      ! d(x)/d(Beta0)
8620      A(K2,1)=DOT(T1,T4)      ! d(y)/d(Beta0)
8630      !
8640      MAT T1=D:2*L
8650      A(K1,2)=DOT(T1,T3)      ! d(x)/d(Beta1)
8660      A(K2,2)=DOT(T1,T4)      !   etc.
8670      !
8680      MAT T1=D:3*L
8690      A(K1,3)=DOT(T1,T3)
8700      A(K2,3)=DOT(T1,T4)
8710      !
8720      MAT T1=D:4*L
8730      A(K1,4)=DOT(T1,T3)
8740      A(K2,4)=DOT(T1,T4)
8750      !
8760  NEXT K
8770  !
8780  DISP
8790  SUBEND

```

Fill-y

The primary purpose of this subroutine is to calculate the column vector of differences between the calculated and measured star coordinates. As input we pass the coordinate frame rotation matrix, and an array containing the direction cosines and measured coordinates for up to five paired stars.

Subroutine Phoeqn is called for each star to compute the image coordinates. These are subtracted from the measured values to fill the deviation vector.

```
*****
*
*           F I L L - Y
*
*****
```

```
7260  ! Fill-y fills Dely(*) with deviation between measured and calc.
7270  ! star positions.
7280  SUB Fill_y(Fld(*),Kk,C(*),Dely(*),Isum)
7290  OPTION BASE 1
7300  DIM Xx(2),Cosines(3)
7310  !
7320  DISP "Fill_y"
7330  !
7340  Idim=Isum+(k*2)
7350  REDIM Dely(Idim)
7360  IF Kk=0 THEN SLBEXIT
7370  !
7380  FOR K=1 TO Kk
7390    FOR I=1 TO 3
7400      Cosines(I)=Fld(K,I)      ! Get direction cosines for this star.
7410    NEXT I
7420    CALL Phoeqn(Cosines(*),C(*),Xx(1),Xx(2))  ! Compute (x,y).
7430    FOR I=1 TO 2
7440      Isum=Isum+1
7450      Dely(Isum)=Fld(K,3+I)-Xx(I) ! Compute deviations from measured (x,y).
7460    NEXT I
7470  NEXT K
7480  !
7490  REDIM Dely(Isum)
7500  !
7510  DISP
7520  SUBEND
```

Gauss

Gauss generates and adds pseudo-Gaussian noise to a variable. The method is to add together 12 random numbers between 0 and 1 and subtract 6. This yields a number from a pseudo-Gaussian distribution with a mean of 0 and a standard deviation of one. Obviously, this distribution will be truncated at ± 6 sigma. This number is then scaled by the standard deviation and added to the mean value (both required as input) and the result returned to the calling program.

IIII

IIII

Kalman

Kalman performs the Kalman filter calculations outlined in Section 4 of the Final Report. The integrated and calculated states and corresponding covariance matrices are passed to Kalman. Currently, we also provide the true state for comparison. The integrated state is replaced by the optimal state on return and the integrated covariance matrix is replaced by the updated matrix.

```
*****
*
*                               K A L M A N
*
*****
```

```
12100 ! Kalman computes the optimal state estimate at each time.
12110 !
12120 ! T. Srikavenda.....28 May 1981.
12130 !
12140 SUB Kalman(Xkt(*),Xki(*),Pk(*),Xkb(*),Lam(*))
12150 OPTION BASE 1
12160 DIM Xk(7),Corr(7)
12170 DIM S1(7,7),S2(7,7)
12180 DIM Kal(7,7),Dev(7)
12190 !
12200 DISP "Kalman"
12210 U1=16
12220 U2=0
12230 !
12240 PRINT USING "/K";"      Kalman Filter State Estimation"
12250 !
12260 MAT S1=TRN(Pk)      !
12270 MAT Pk=S1+Pk      ! Ensure symmetric Pk.
12280 MAT Pk=(.5)+Pk    !
12290 !
12300 FOR I=3 TO 7
12310   Xkb(I)=Xki(I)    ! Fill measurement vector with bias values.
12320 NEXT I
12330 !
12340 MAT S1=Lam+Pk      ! Add obs. cov. matrix to integrated cov. matrix.
12350 !
12360 MAT S2=TRN(S1)     !
12370 MAT S1=S1+S2      ! Ensure symmetric matrix.
12380 MAT S1=(.5)+S1    !
12390 !
12400 MAT S2=INV(S1)
12410 !
12420 MAT S1=TRN(S2)     !
12430 MAT S2=S1+S2      ! Ensure symmetric matrix.
12440 MAT S2=(.5)+S2    !
12450 !
12460 MAT Kal=Pk*S2      ! Compute Kalman gain matrix.
12470 !
12480 MAT Dev=Xkb-Xki    ! Deviations between states--(Obser.-integ. state).
12490 !
12500 MAT Corr=Kal*Dev    ! Correction vector.
12510 MAT Xk=Xki+Corr    ! Calc. optimal state.
12520 MAT S1=Kal*Pk      ! Calc. updated covariance matrix.
12530 MAT Pk=Pk-31
12540 !
12550 MAT S1=TRN(Pk)     !
12560 MAT Pk=Pk+31      ! Ensure symmetric matrix.
```

```

12570 MAT Pk:=(.5)+Pk      !
12580      !
12590 PRINT USING "/K";"   _____ State Vectors _____
      Difference:         "
12600 PRINT USING "K";"    True      Integ.      Proc. B      Opt. Est.      (I-T)
      (B-T)      (O-T)'
12610      !
12620 FOR I=1 TO 7
12630   PRINT USING Form2;Xkt(I),Xki(I),Xkb(I),Xk(I),Xki(I)-Xkt(I),Xkb(I)-Xkt(I)
      ,Xk(I)-Xkt(I)
12640 NEXT I
12650      !
12660 REDIM Xk(4)
12670 Norm=SQR(DJT(Xk,Xk))
12680 PRINT USING Form1;"Norm of optimal estimate - 1:";Norm-1
12690 MAT Xk=Xk/(Norm) ! Normalize the optimal estimate.
12700 REDIM Xk(7)
12710      !
12720 MAT Xki=Xk          ! Set state = optimal state...This is the starting
12730                        ! estimate for the next frame.
12740      !
12750 Form1: (IMAGE /K),MD.DDDDDDDDD/
12760 Form2: (IMAGE 7(MD.DDDE,X)
12770 Form3: (IMAGE 7(MD.DDDE,X)/
12780      !
12790 DISP
12800 SUBEND

```

Least

Least solves the least-squares problem for differential corrections to the Euler parameters. Formally, the solution is:

$$\Delta\beta = (A^T W A)^{-1} A^T W \Delta X$$

where W is a weight matrix, A is a matrix containing partial derivatives of image coordinates with respect to Euler parameters, and ΔX is a vector of differences between measured and calculated coordinates.

However, we have adopted a diagonal weight matrix and thus absorb the weights into A and ΔX (in the calling program). Therefore, we write

$$\Delta\beta = (A^T A)^{-1} A^T \Delta X,$$

the form used in this subroutine.

The covariance-like result $(A^T A)^{-1}$ is returned to the calling program along with the corrections, $\Delta\beta$.

```

*****
*
*               L E A S T
*
*****

```

```

7610 ! This routine computes  $((A[transpose] * A)[inverse]) * A[transpose] * Dy$ 
7620 ! where Dy is the vector of deviations.
7630 SUB Least( $\beta(*)$ ,  $Dx(*)$ ,  $Cov(*)$ ,  $Dy(*)$ , Idim, Jdim)
7640 OPTION BASE 1
7650 DIM T1(Jdim, Jdim), T2(Jdim), At(Jdim, Idim)
7660 !
7670 DISP "Least"
7680 !
7690 MAT At=TRN(A)
7700 MAT T1=At* $\beta$ 
7710 MAT Cov=INV(T1) ! Cov =  $(A[transpose] * A)[inverse]$ .
7720 MAT T2=At*Dy
7730 MAT Dx=Cov*T2
7740 !
7750 DISP
7760 SUBEND

```

Least-1

This subroutine performs an iterative least-squares correction using coordinate data from one field of view. The current values of β_{VN} are passed to Least-1, along with the rotation matrix AV or BV and the matrix containing direction cosines and measured image coordinates for paired stars for that field of view. The corrected Euler parameters are returned along with the covariance matrix result from the least-squares correction.

The method employed is to first call Dircosb to compute the rotation matrix VN and its partial derivatives. Fill-y then computes the differences between measured and calculated image coordinates. Pre-mult converts partial derivatives of VN to derivatives of AN or BN by multiplying by AV or BV, respectively. Dxdbeta uses these partials to compute the partials of image coordinates with respect to Euler parameters, β_{VN} . The last row of the derivative matrix is filled with the constraint equation.

Finally, subroutine Least is called to compute corrections to β_{VN} . If these are small enough we return to the calling program. Otherwise, we iterate again, up to a limit of six times.

```

*****
*
*           L E A S T - 1
*
*****

11480 SUB Least_1(Bun(*),Av(*),Fld(*),Kmax,W,Cov(*),Converge)
11490 OPTION BASE 1
11500 DIM Saveb(4),Bs(4),Dun1(3,3),Dun2(3,3),Dun3(3,3),Dun4(3,3),An(3,3),Vn(3,3)
11510 DIM Delx(4),Dely(11),A(11,4)
11520 !
11530 MAT Saveb=Bun ! Save the original values of Euler parameters.
11540 !
11550 PRINT USING "/K";" Least-Squares Correction For One FOV"
11560 FOR It=1 TO 6
11570 CALL DirCosb(Bun(*),Vn(*),1,Dun1(*),Dun2(*),Dun3(*),Dun4(*))
11580 MAT An=Av+Vn
11590 Isum=0
11600 REDIM Dely(Kmax*2)
11610 CALL Fill_y(Fld(*),Kmax,An(*),Dely(*),Isum)
11620 Sq1=3QR(DOT(Iely,Dely)/(Kmax*2))
11630 PRINT USING "/,K,D.DDDDE";"RMS error for normalized image coordinates:",
Sq1
11640 !
11650 REDIM A(max*2+1,4),Dely(2*Kmax+1)
11660 CALL Pre_mult(Av(*),Dun1(*),Dun2(*),Dun3(*),Dun4(*))
11670 CALL DxdBeta(Fld(*),A(*),Kmax,An(*),Dun1(*),Dun2(*),Dun3(*),Dun4(*))
11680 !
11690 FOR I=1 TO 4
11700 A(2*Kmax+1,I)=2*W*Bun(I) ! Constraint equation.
11710 NEXT I
11720 !
11730 Dely(2*Kmax+1)=W*(1-DOT(Bun,Bun))
11740 !
11750 CALL Least(A(*),Delx(*),Cov(*),Dely(*),Kmax*2+1,4)
11760 !
11770 MAT Bs=Bun
11780 PRINT USING "/,K";" Beta(old) Beta(new) Delta(Beta)"
11790 FOR I=1 TO 4
11800 Bun(I)=Bun(I)+Delx(I)
11810 PRINT USING "X3(MD.DDDDDDD,XXXX)";Bs(I),Bun(I),Delx(I)
11820 NEXT I
11830 Dev=3QR(DOT(Ielx,Delx)/4)
11840 PRINT USING "/,K,D.DDDDE";"RMS change in Euler parameters: ",Dev
11850 !
11860 IF Dev<1E-6 THEN Morestars ! Small corrections...exit loop.
11870 NEXT It
11880 !
11890 Nosoln: PRINT "***** LEAST-SQUARES FOR ONE FOV DID NOT CONVERGE *****"
11900 Converge=0 ! Failure of least-squares.
11910 MAT Bun=Saveb ! Replace Euler parameters with original values.
11920 MAT Cov=ZER
11930 PRINT USING Form1;"Number of iterations:",It-1,Converge

```

```
11940 SUBEXIT
11950 |
11960 Morestars: Converge=i
11970 MAT Cov=Cov*(Sq1*Sq1) | Compute cov. matrix...mult. by sigma**2.
11980 PRINT USING Form1;"Number of iterations:",It,Converge
11990 Form1: IMAGE K,X,DD,XXX,"Converge=",D
12000 |
12010 SUBEND
```

Least-2

Least-2 updates the Euler parameters β_{VN} and β_{BA} via least-squares differential correction, using between 3 and 5 matched stars from each FOV. Both β_{VN} and β_{BA} should initially be very near their final corrected values (β_{VN} has been corrected by Pair-It and β_{BA} does not vary rapidly); thus the least-squares requires only 2 or 3 iterations to converge. Since this also means the derivatives do not vary substantially between iterations, we can use the secant method (subroutine Secant) to update the matrix of partial derivatives used in the least-squares.

As input we pass the current values of β_{VN} and β_{BA} and the arrays containing the direction cosines and measured coordinates for up to 5 stars per FOV. We return the updated Euler parameters β_{VN} and β_{BA} and the 4×4 covariance matrix associated with β_{VN} . For each least-squares iteration we compute the differences between measured and calculated images for corresponding stars and the change in these differences compared with the previous iteration (to be used by Secant). On the first iteration we calculate the exact derivatives of image coordinates with respect to both β_{VN} and β_{BA} (via calls to Dircosb, Mat-av, Pre-mult, Post-mult and Dxdbeta - see Appendix 7). The last several rows of the derivative matrix are filled with the constraint equations, one for each set of Euler parameters (multiplied by an appropriate weight) β_{BA} , see Section 3 and Appendix 7). A call to Least returns corrections to all eight Euler parameters; if these are small we return, otherwise we iterate again, using Secant to update derivatives.

```
*****
*
*           L E A S T - 2
*
*****
```

```
15180 SUB Least_2(Bvr(*),Bba(*),Ka,Flda(*),Kb,Fldb(*),W,Cov8(*))
15190 OPTION BASE 1
15200 DIM A(26,8),Dely(26),Ddy(26),At1(10,4),At2(10,4),Delx(8),Bbasave(4)
15210 DIM An(3,3),Bn(3,3)
15220 DIM Av(3,3),Dav1(3,3),Dav2(3,3),Dav3(3,3),Dav4(3,3)
15230 DIM Ba(3,3),Dbal(3,3),Dbal2(3,3),Dbal3(3,3),Dbal4(3,3)
15240 DIM Vn(3,3),Dvn1(3,3),Dvn2(3,3),Dvn3(3,3),Dvn4(3,3)
15250 !
15260 PRINT USING "/,K";"   Least-Squares Correction For Two FOV"
15270 PRINT USING "/,K,D,K,D,K";"Correct orientation using ",Ka," stars from FOV
(A) and ",Kb," stars from FOV(B)."
```

15280 SqtS=1E40

15290 Ka2=Ka+Ka

15300 Kb2=Kb+Kb

15310 Kk=Ka2+Kb2

15320 Ipass2=1

15330 Jdim=8

15340 Idim=Ka2+Kb2+2+4

15350 !

15360 REDIM A(Idim,Jdim),Ddy(Idim)

15370 MAT Dely=ZER

15380 MAT A=ZER

15390 MAT Cov8=ZER

15400 MAT Bbasave=Bba

15410 !

15420 !

15430 FOR It=1 TO 4

15440 Exact: !

15450 Isum=0

15460 CALL Dirccsb(Bvn(*),Vn(*),Ipass2,Dvn1(*),Dvn2(*),Dvn3(*),Dvn4(*))

15470 CALL Dirccsb(Bba(*),Ba(*),Ipass2,Dbal(*),Dbal2(*),Dbal3(*),Dbal4(*))

15480 CALL Mat_av(Ba(*),Dbal(*),Dbal2(*),Dbal3(*),Dbal4(*),Av(*),Ipass2,Dav1(*),Dav2(*),Dav3(*),Dav4(*))

15490 MAT A1=Av*Vn

15500 MAT B1=Ba*An

15510 REDIM Dcy(Kk),Dely(Kk)

15520 MAT Ddy=Dely

15530 CALL Fill_y(Flda(*),Ka,An(*),Dely(*),Isum) ! Deviations for FOV(A).

15540 CALL Fill_y(Fldb(*),Kb,Bn(*),Dely(*),Isum) ! Deviations for FOV(B).

15550 MAT Ddy=Dcy-Dely

15560 Sqt=SQR(DCT(Dely,Dely)/Kk)

15570 PRINT USING "/,K,X,D.DDDDE";"RMS error in normalized image coordinate

s: ";Sqt

15580 IF Sqt<SqtS THEN Decreasing

15590 Ipass2=1 ! Flag to compute exact derivs. because the

15600 SqtS=1E40 ! image error is increasing.

15610 GOTD Exact

```

15620      !
15630 Decreasing:      ! Come here is solution is converging.
15640      Sqts=3qt
15650      REDIM ldy(Idim)
15660      IF Ipass2=2 THEN Secant_method
15670      Ipass2=2 !
15680      ! Compute partial derivus. of (x,y) for FOV(A).
15690      CALL Pre_mult(Av(*),Dun1(*),Dun2(*),Dun3(*),Dun4(*))
15700      CALL Post_mult(Vn(*),Dav1(*),Dav2(*),Dav3(*),Dav4(*))
15710      CALL Dxdbeta(Flda(*),At1(*),Ka,Rn(*),Dun1(*),Dun2(*),Dun3(*),Dun4(*))
15720      CALL Dxdbeta(Flda(*),At2(*),Ka,Rn(*),Dav1(*),Dav2(*),Dav3(*),Dav4(*))
15730      !
15740      FOR I=1 TO 4
15750          FOR K=1 TO Ka2
15760              A(K,I)=At1(K,I)      ! Fill A(*) with FOV(A) derivatives.
15770              A(K,I+4)=At2(K,I)
15780          NEXT K
15790      NEXT I
15800      !
15810      ! Compute partial derivus. of (x,y) for FOV(B).
15820      CALL Pre_mult(Ba(*),Dun1(*),Dun2(*),Dun3(*),Dun4(*))
15830      CALL Post_mult(Rn(*),Dba1(*),Dba2(*),Dba3(*),Dba4(*))
15840      CALL Pre_mult(Ba(*),Dav1(*),Dav2(*),Dav3(*),Dav4(*))
15850      MAT Dba1=Iba1+Dav1
15860      MAT Dba2=Iba2+Dav2
15870      MAT Dba3=Iba3+Dav3
15880      MAT Dba4=Iba4+Dav4
15890      CALL Dxdbeta(FlDb(*),At1(*),Kb,Rn(*),Dun1(*),Dun2(*),Dun3(*),Dun4(*))
15900      CALL Dxdbeta(FlDb(*),At2(*),Kb,Rn(*),Dba1(*),Dba2(*),Dba3(*),Dba4(*))
15910      !
15920      FOR I=1 TO 4
15930          FOR K=1 TO Kb2
15940              A(Ka2+K,I)=At1(K,I)      ! Fill A(*) with FOV(B) derivatives.
15950              A(Ka2+K,I+4)=At2(K,I)
15960          NEXT K
15970      NEXT I
15980      !
15990      GOTO Lsq
16000 Secant_method: !
16010      CALL Secant(A(*),Delx(*),Ddy(*),Idim,Jdim)
16020 Lsq: !
16030      REDIM ldy(Idim)
16040      !
16050      FOR I=1 TO 4
16060          A(K+1,I)=Bun(I)*W*2      ! Constraint eq. for Bun.
16070          A(K+1,4+I)=0
16080          A(K+2,4+I)=Bba(I)*W*2      ! Constraint eq. for Bba.
16090          A(K+2,I)=0
16100      NEXT I
16110      !
16120      Dely((K+1))=W*(1-DOT(Bun,Bun))      ! More constraint eq.
16130      Dely((K+2))=W*(1-DOT(Bba,Bba))      ! " " "
16140      MAT Ddy=Dely
16150      !
16160      CALL Least(A(*),Delx(*),CovB(*),Dely(*),Idim,Jdim)

```

```

16170      !
16180      PRINT USING "/,K/K";"      Euler parameters and corrections:", "      B
(V-N)  delta-B(V-N)      B(B-A)      delta-B(B-A)"
16190      !
16200      FOR I=1 TO 4
16210          Bun(I)=Eun(I)+Delx(I)
16220          Bba(I)=Eba(I)+Delx(4+I)
16230          PRINT USING "4(MD.DDDDDDD,XX)"; Bun(I); Delx(I); Bba(I); Delx(4+I)
16240      NEXT I
16250      !
16260      Dev=SQR(DCT(Delx, Delx)/8)
16270      PRINT USING "/,K,X,D.DDDDE"; "RMS change in Euler parameters:"; Dev
16280      IF Dev<1E-7 THEN Covariance
16290      NEXT It
16300      GOTO Two_failed
16310      !
16320      Covariance: !
16330      PRINT "      (End of least-squares for two FOV)"
16340      MAT Cov8=CovE*(Sqr^2)      !Mult.(A[transpose]*A)(inverse) by sigma^2.
16350      ! PRINT USING "8(MD.DDE,X)"/"; Cov8(*)
16360      SUBEXIT
16370      Two_failed:      !
16380      PRINT "-->-->-->--> LEAST-2 FAILED <--<--<--<--<--"
16390      !
16400      SUBEND

```

Mata

This subroutine computes part of the right hand side of kinematic differential equations governing Euler parameters. For input data Mata requires the current Euler parameters, β_{VN} , the rate gyro data and the rotation matrix VG to rotate the gyro rates from the gyro to vehicle frame. We can express the differential equations as

$$\begin{aligned}\{\dot{\beta}\} &= [\omega]\beta \\ &= [\beta]\omega\end{aligned}$$

(see Section 4 and Appendix 8 of the Final Report for details); Mata fills matrices $[\omega]$ and $[\beta]$. These two forms are also needed for integrating the matrix Riccati equation for covariance propagation.

```
*****  
*                                     *  
*               M A T R              *  
*                                     *  
*****  
  
12890 SUB Matra(F,W(*),X(*),A11(*),A12(*) )  
12900 OPTION BASE 1  
12910 COM Vg(3,3)  
12920 DIM A12p(4,3),Wv(3)  
12930      !  
12940 MAT Wv=Wg*.4          ! Rotate gyro rates into V frame.  
12950 MAT Wv=Wv*(.5)       ! Divide by 2 now instead of later.  
12960 W1=Wv(1)  
12970 W2=Wv(2)  
12980 W3=Wv(3)  
12990      !  
13000      !   Calc. matrix A11=D<<B0,B1,B2,B3>DOT>/D(B0,B1,B2,B3)  
13010      !                               where B0,B1,B2,B3 are Euler parameters.  
13020      !  
13030 A11(1,1)=0  
13040 A11(1,2)=-.41  
13050 A11(1,3)=-.42  
13060 A11(1,4)=-.43  
13070      !  
13080 A11(2,1)=W1  
13090 A11(2,2)=0  
13100 A11(2,3)=W3  
13110 A11(2,4)=-.42  
13120      !  
13130 A11(3,1)=W2  
13140 A11(3,2)=-.43  
13150 A11(3,3)=0  
13160 A11(3,4)=W1  
13170      !  
13180 A11(4,1)=W3  
13190 A11(4,2)=W2  
13200 A11(4,3)=-.41  
13210 A11(4,4)=0  
13220      !  
13230 IF F=0 THEN SUEEXIT  
13240      !  
13250      !   Calc. matrix A12 = -D<<B0,B1,B2,B3>DOT>/D(W1,W2,W3)  
13260      !                               = D<<B0,B1,B2,B3>DOT>/D(b1,b2,b3)  
13270      !  
13280 B0=X(1)*.5  
13290 B1=X(2)*.5  
13300 B2=X(3)*.5  
13310 B3=X(4)*.5  
13320      !  
13330 A12p(1,1)=B1  
13340 A12p(1,2)=B2  
13350 A12p(1,3)=B3
```

```
13360      !
13370 A12p(2,1)=-B0
13380 A12p(2,2)=B3
13390 A12p(2,3)=-B2
13400      !
13410 A12p(3,1)=-B3
13420 A12p(3,2)=-B0
13430 A12p(3,3)=B1
13440      !
13450 A12p(4,1)=B2
13460 A12p(4,2)=-B1
13470 A12p(4,3)=-B0
13480      !
13490 MAT A12=A12p*Vc      ! Multiply by rotation matrix.
13500      !
13510 SUBEND
```

Mat-av

This subroutine calculates the rotation matrix, AV , between the vehicle and star tracker (A) frame. The matrix, BA , between star tracker frames is required for input. If selected, the partial derivatives of AV with respect to elements of BA are calculated. (See Appendix 7 for details).

```

*****
*
*           M A T _ A V
*
*****

10220 SUB Mat_av(Ba(*),Dba0(*),Dba1(*),Dba2(*),Dba3(*),Av(*),Der,Dav0(*),Iav1(*)
,Dav2(*),Dav3(*))
10230 OPTION BASE 1
10240 DIM T1(3,3),T2(3,3),T3(3,3),T4(3,3)
10250 !
10260 DISP "Mat_av"
10270 !
10280 D1=1/SQR(2+2*Ba(3,3))      ! Two useful factors.
10290 D2=1/SQR(2-2*Ba(3,3))
10300 !
10310 Av(1,1)=Ba(3,1)*D1        ! Compute matrix AV from BA.
10320 Av(2,1)=Ba(3,2)*D1
10330 Av(3,1)=.5/D1
10340 !
10350 Av(1,2)=Ba(3,1)*D2
10360 Av(2,2)=Ba(3,2)*D2
10370 Av(3,2)=-.5/D2
10380 !
10390 Av(1,3)=-2*Ba(3,2)*D1*D2
10400 Av(2,3)=2*Ba(3,1)*D1*D2
10410 Av(3,3)=0
10420 !
10430 DISP
10440 !
10450 IF Der<>1 THEN SUBEXIT      ! Leave SUB if we don't need partials.
10460 !
10470 DISP "Mat_av"
10480 !
10490 MAT T1=ZER                  ! Compute d(AV(*)/d(BA(3,1)).
10500 T1(1,1)=D1
10510 T1(1,2)=D2
10520 T1(2,3)=2*D1*D2
10530 !
10540 MAT T2=ZER                  ! Compute d(AV(*)/d(BA(3,2)).
10550 T2(2,1)=T1(1,1)
10560 T2(2,2)=T1(1,2)
10570 T2(1,3)=-T1(2,3)
10580 !
10590 T3(1,1)=-Av(1,1)*D1*D1      ! Compute d(AV(*)/d(BA(3,3)).
10600 T3(2,1)=-Av(2,1)*D1*D1
10610 T3(3,1)=.5*D1
10620 T3(1,2)=Av(1,2)*D2*D2
10630 T3(2,2)=Av(2,2)*D2*D2
10640 T3(3,2)=D2*.5
10650 T=Ba(3,3)+3*(D1*D2)^3
10660 T3(1,3)=-Ba(3,2)*T
10670 T3(2,3)=Ba(3,1)*T

```

```

10680 T3(3,3)=0
10690      !
10700 MAT Dav0=T1+(Dta0(3,1))  ! Compute d(AV(*) )/d(Bba(1)).
10710 MAT T4=T2*(Iba0(3,2))
10720 MAT Dav0=Dav0+T4
10730 MAT T4=T3*(Iba0(3,3))
10740 MAT Dav0=Dav0+T4
10750      !
10760 MAT Dav1=T1+(Dta1(3,1))  ! Compute d(AV(*) )/d(Bba(2)).
10770 MAT T4=T2*(Iba1(3,2))
10780 MAT Dav1=Dav1+T4
10790 MAT T4=T3*(Iba1(3,3))
10800 MAT Dav1=Dav1+T4
10810      !
10820 MAT Dav2=T1+(Dta2(3,1))  ! Compute d(AV(*) )/d(Bba(3)).
10830 MAT T4=T2*(Iba2(3,2))
10840 MAT Dav2=Dav2+T4
10850 MAT T4=T3*(Iba2(3,3))
10860 MAT Dav2=Dav2+T4
10870      !
10880 MAT Dav3=T1+(Dta3(3,1))  ! Compute d(AV(*) )/d(Bba(4)).
10890 MAT T4=T2*(Iba3(3,2))
10900 MAT Dav3=Dav3+T4
10910 MAT T4=T3*(Iba3(3,3))
10920 MAT Dav3=Dav3+T4
10930      !
10940 DISP
10950 SUBEND

```

Orbit

We use Herrick's "f and g" solution for a two body case (see Reference 6) to update the satellite and earth position and velocities. For either the satellite or earth we set:

$$r_0 = (\underline{r}_0 \cdot \underline{r}_0)^{1/2} \quad (X_0, Y_0, Z_0) = \text{initial position}$$

$$= (X_0^2 + Y_0^2 + Z_0^2)^{1/2}$$

$$V_0 = (\underline{V}_0 \cdot \underline{V}_0)^{1/2} = (\dot{\underline{r}}_0 \cdot \dot{\underline{r}}_0)^{1/2} \quad (\dot{X}_0, \dot{Y}_0, \dot{Z}_0) = \text{initial velocity}$$

$$= (\dot{X}_0^2 + \dot{Y}_0^2 + \dot{Z}_0^2)^{1/2}$$

$$D_0 = \underline{r}_0 \cdot \underline{V}_0$$

$$= X_0 \dot{X}_0 + Y_0 \dot{Y}_0 + Z_0 \dot{Z}_0$$

$$1/a = 2/r_0 - v_0^2/u$$

$$u = GM$$

$$t_0 = \text{initial time}$$

In the current version of this program we have set both orbits to be circular. The inclinations of the earth orbit is 23.5° and the satellite orbit is 70°.

To obtain the position and velocity of either body at some later time, t , we solve the following equation for M by Newton's method:

$$u^{1/2}(t - t_0)a^{-3/2} = M - (1 - r_0/a)\sin M + D_0(1 - \cos M)(ua)^{1/2}$$

using the initial estimate:

$$M = u^{1/2}(t - t_0)a^{-3/2}.$$

Then, the position (X, Y, Z) at time t is:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = f \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix} + g \begin{pmatrix} \dot{X}_0 \\ \dot{Y}_0 \\ \dot{Z}_0 \end{pmatrix}$$

where

$$f = 1 - a(1 - \cos M)/r_0$$

$$g = (t - t_0) - a^{3/2}(M - \sin M)u^{-1/2}.$$

Also, the velocity $(\dot{X}, \dot{Y}, \dot{Z})$ at time t is:

$$\begin{pmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{pmatrix} = \dot{f} \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix} + \dot{g} \begin{pmatrix} \dot{X}_0 \\ \dot{Y}_0 \\ \dot{Z}_0 \end{pmatrix}$$

where $r = (X^2 + Y^2 + Z^2)^{1/2}$

$$\dot{f} = -(ua)^{1/2} \sin M / rr_0$$

$$\dot{g} = 1 - a(1 - \cos M)/r.$$

```

*****
*
*               O R B I T
*
*****

```

```

16490 ! This subroutine computes the orbital position and velocity using
16500 ! Herrick's F and G solution. (See J.L.Junkins Text)
16510 !
16520 SUB Orbit(T,P(*),V(*),P0(*),V0(*),T0,U,A,D0,R0)
16530 OPTION BASE 1
16540 DIM P1(3),P2(3)
16550 !
16560 X=1-R0/A
16570 Y=D0/SQR(U*A)
16580 Rho=SQR(U)*(T-T0)/SQR(A*A*A)
16590 Phi=Rho
16600 Dphi=10
16610 !
16620 FOR I=1 TO 10 ! Find Phi by Newton's method.
16630 Cphi=COS(Phi)
16640 Sphi=SIN(Phi)
16650 IF ABS(Dphi)<1E-5 THEN Got_it
16660 Rhoc=Phi-X*Sphi+Y*(1-Cphi)
16670 Drdp=1-X*Cphi+Y*Sphi
16680 Dphi=(Rho-Rhoc)/Drdp
16690 Phi=Phi+Dphi
16700 NEXT I
16710 !
16720 PRINT '***** HELP ***** ORBIT DID NOT FIND PHI !!!!!!!!!!!'
16730 !
16740 Got_it: ! Newton's method worked.
16750 F=1-A*(1-Cphi)/R0
16760 G=T-T0-A*SQR(A/U)*(Phi-Sphi)
16770 MAT P1=(F)*P0 !
16780 MAT P2=(G)*V0 ! Update position.
16790 MAT P=P1+P2 !
16800 !
16810 R=SQR(DOT(P,P))
16820 Fd=-SQR(U*A)*Sphi/(R*R0)
16830 Gd=1-A*(1-Cphi)/R
16840 MAT P1=(Fd)*P0 !
16850 MAT P2=(Gd)*V0 ! Update velocity.
16860 MAT V=P1+P2 !
16870 !
16880 SUBEND

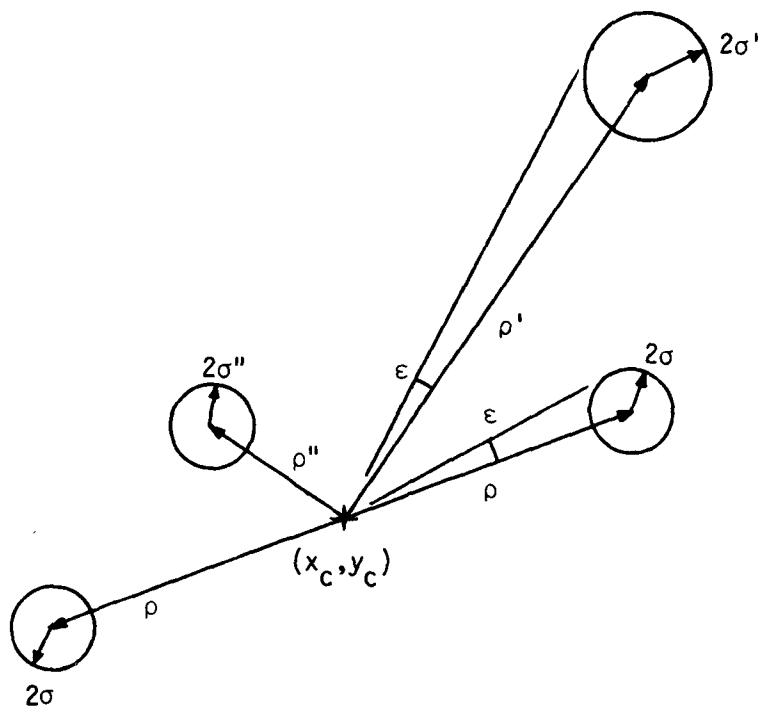
```

Pair-it

The task of Pair-it is to identify measured stars with specific catalog stars. We do this in two ways. The first method compares the cosine of the interstar angle between measured star pairs with the cosine of interstar angles of catalog star pairs. If there is a match we perform a least-squares correction (by calling Least-1) to refine the attitude estimate.

The second method to match stars uses the improved attitude estimate to mathematically project all the sub-catalog stars onto the focal plane and then compare each position with measured stars. We require at least three matches to confirm the attitude found by least-squares. The direction cosines and measured coordinates for each confirmed star image (up to 5 stars) are stored in an array and returned to the calling program for later processing.

The confirmation tests discussed above require an error tolerance between projected and measured stars in order to accept or reject a specific catalog star. The technique we use is discussed below. We first calculate, for one star of the initial matched pair, the angular size of twice the estimated one-sigma error (10% pixel) as seen from the midpoint between images and perpendicular to the line connecting the two stars. Then, for stars more distant than one-half the pair separation, we scale the angle by the distance from the midpoint in order to get the tolerance for each star. For nearer stars we simply use the estimated two-sigma image error. This technique helps to account for rotation errors due to displacements of one or both stars of the initial pair, normal to the separation vector. (See Figure A10.3).



Let: $\tan \epsilon \approx \epsilon \approx 2\sigma/\rho$

where ρ = one-half of separation between the initial matched pair,

and σ = estimated error (1-sigma) in position of star centroid.

Then: $2\sigma' = \epsilon\rho'$ (for $\rho' > \rho$)

$2\sigma'' = 2\sigma$ (for $\rho'' \leq \rho$)

where ρ' and ρ'' are distances from midpoint (x_c, y_c) between stars of initial pair.

Figure A10.3: Calculation of error tolerance values to be used for matching measured and calculated star images.

```

*****
*
*               P A I R - I T
*
*****

```

```

4620  ! Pair_it pairs stars from the catalog and stars from Proc. A to find
4630  ! the orientation of the vehicle.
4640  SUB Pair_it(SHRT Fovc(*),REAL Nfovc,Fovm(*),Nfovm,Bvn(*),Av(*),Tol,Sigxy,
F1,W,Fld(*),Kmax)
4650  OPTION BASE 1
4660  DIM Vn(3,3),An(3,3),Sb(4)
4670  DIM Xx(2),Dist(5),L(3),Cov(4,4),T(3,3)
4680  SHORT Cosm(45),Epsp(45)
4690  INTEGER Indx(45,2)
4700  RAD
4710  !
4720  MAT Sb=Bvn  ! Save original values of Euler parameters.
4730  Cosmax=COS(PI/180)
4740  Cosmin=COS(11*PI/180)
4750  L1=1
4760  !
4770  Mm=Nfovm+Nfovm-1  ! Maximum sum of indices for measured stars.
4780  PRINT USING "/,K";"Table of Cos(Theta) for Measured Stars."
4790  PRINT USING Form2
4800  Form2: IMAGE "  Star Star  Cos(Theta)"
4810  !
4820  FOR M=3 TO Mm      ! Loop over all possible sums of indices for
4830      J1=(M-1)/2      ! measured stars.
4840      FOR J=1 TO J1    ! Loop over all pairs whose indices sum to M.
4850          K=M-J
4860          IF K>Nfovm THEN GOTO Nextj
4870          Cosm(L1)=Fovm(J,3)*Fovm(K,3)*(Fovm(J,1)*Fovm(K,1)+Fovm(J,2)*Fovm(K,2)+
1)
4880          IF Cosm(L1)>Cosmax THEN GOTO Nextj
4890          PRINT USING Form1;J;K;Cosm(L1)
4900  Form1: IMAGE 2X,DDD,2X,DDD,3X,D.DDDDDD
4910          Indx(L1,1)=J
4920          Indx(L1,2)=K
4930          L1=L1+1
4940  Nextj: NEXT J
4950      NEXT M
4960      Imax=L1-1      ! Total number in list.
4970      !
4980  Mm=Nfovc+Nfovc-1  ! Maximum sum of indices for catalog stars.
4990  PRINT USING "/,K";"Begin Pairing Catalog Stars and Comparing To Measured P
airs."
5000  PRINT USING "/,K,X,DDD";"Number of stars from catalog:",Nfovc
5010  PRINT USING Form2
5020  !
5030  FOR M=3 TO Mm      ! Loop over all possible sums of indices of
5040      J1=(M-1)/2      ! catalog stars.

```

```

5050     FOR Jj=1 TC J1 ! Loop over all pairs whose indices sum to M.
5060     K=M-Jj
5070     IF K>Nfoc THEN GOTO Nextjj
5080     Cost=0
5090     FOR L=1 TO 3
5100         Cost=Cst+Fovc(Jj,L)*Fovc(K,L) ! Compute dot product.
5110     NEXT L
5120     PRINT USING Form1;Jj;K;Cost
5130     IF Cost>Cosmax THEN GOTO Nextjj
5140     IF Cost<Cosmin THEN GOTO Nextjj
5150     FOR Ii=1 TO Imax
5160         IF ABS(Cost-Cosm(Ii))<Tol THEN Match ! Test for match.
5170 Nextii:     NEXT Ii
5180 Nextjj:     NEXT Jj
5190     NEXT M
5200     !
5210 Failed: PRINT "***** NO PAIR MATCH FOUND FOR THIS FOV *****"
5220     Kmax=0
5230     SUBEXIT
5240     !
5250 Match: PRINT USING "/,K";">>>>> Catalog Pair Matched with Measured Pair <
<<<<"
5260     Kmax=2
5270     Im1=Incx(Ii,1)
5280     Im2=Incx(Ii,2)
5290     Ic1=Jj
5300     Ic2=K
5310     IF Fovm(Im1,4)<Fovm(Im2,4) THEN Okm ! Test magnitude order.
5320     Is=Im1
5330     Im1=Im2 ! Switch magnitude.
5340     Im2=Is
5350 Okm: IF Fovc(Ic1,4)<Fovc(Ic2,4) THEN Okc ! Test magnitude order.
5360     Is=Ic1
5370     Ic1=Ic2 ! Switch magnitude.
5380     Ic2=Is
5390 Okc: PRINT USING "/K";"Measured pair:"
5400     PRINT USING Form2
5410     PRINT USING Form1;Im1;Im2;Cosm(Ii)
5420     !
5430     ! Compute separation of pair/2.
5440     !
5450     Rho12=SQR((Fovm(Im1,1)-Fovm(Im2,1))^2+(Fovm(Im1,2)-Fovm(Im2,2))^2)/2
5460     !
5470     ! Compute angular extent of 2*sigma error.
5480     !
5490     Eps=2*Sigx/(F1*Rho12)
5500     Xcent=(Fovm(Im1,1)+Fovm(Im2,1))/2 ! Compute average position.
5510     Ycent=(Fovm(Im1,2)+Fovm(Im2,2))/2
5520     !
5530     Fld(1,1)=Fovc(Ic1,1)
5540     Fld(1,2)=Fovc(Ic1,2)
5550     Fld(1,3)=Fovc(Ic1,3)
5560     Fld(1,4)=Fovm(Im1,1)
5570     Fld(1,5)=Fovm(Im1,2) !
5580     Fld(2,1)=Fovc(Ic2,1) ! Fill array with data for paired stars.

```

```

5590 Fld(2,2)=Fouc(Ic2,2)      |
5600 Fld(2,3)=Fouc(Ic2,3)
5610 Fld(2,4)=Fouv(Im2,1)
5620 Fld(2,5)=Fouv(Im2,2)
5630 |
5640 | Perform least-squares differential correction.
5650 |
5660 CALL Least_1(Bun(*),Av(*),Fld(*),Kmax,W,Cov(*),Converge)
5670 IF Converge=0 THEN Nextii | Try another pair--this one didn't work.
5680 |
5690 | Search for confirming stars.
5700 |
5710 CALL Dircost(Bun(*),Vn(*),0,T(*),T(*),T(*),T(*))
5720 MAT An=Av*Vn
5730 Kmaxs=Kmax
5740 Kmax=0
5750 MAT Fld=ZER
5760 PRINT USING "/,K"; "          Test For Additional Stars"
5770 PRINT USING "K"; "x(calc)  x(meas)  y(calc)  y(meas)      Dx      Dy"
5780 |
5790 FOR J=1 TO Nfouv
5800 | Compute maximum deviation allowed for each measured star.
5810 Rho=SQR((Fouv(J,1)-Xcent)^2+(Fouv(J,2)-Ycent)^2)
5820 Epsp(J)=1/AX(Eps*Rho,2*Sigxy/F1)
5830 NEXT J
5840 |
5850 FOR N=1 TO Nfouc
5860   FOR Is=1 TO 3
5870     L(Is)=Fouc(N,Is)
5880   NEXT Is
5890   CALL Pncqr(L(*),An(*),Xx(1),Xx(2))
5900   FOR J=1 TO Nfouv
5910     R1=ABS(Xx(1)-Fouv(J,1)) | Deviation in x.
5920     IF R1>Epsp(J) THEN Next_j
5930     R2=ABS(Xx(2)-Fouv(J,2)) | Deviation in y.
5940     IF R2>Epsp(J) THEN Next_j
5950     | Confirming star found.
5960     PRINT USING "6(MD.DDDD,XX)"; Xx(1)*F1,Fouv(J,1)*F1,Xx(2)*F1,Fouv(J,
2)*F1,R1*F1,R2*F1
5970     | PRINT USING "2(MD.DDDD,XX)"; R1*F1,Epsp(J)*F1,R2*F1,Epsp(J)*F1
5980     Kmax=Kmax+1
5990     Fld(Kmax,1)=Fouc(N,1)
6000     Fld(Kmax,2)=Fouc(N,2)
6010     Fld(Kmax,3)=Fouc(N,3) | Save data.
6020     Fld(Kmax,4)=Fouv(J,1)
6030     Fld(Kmax,5)=Fouv(J,2)
6040     |
6050     IF Kmax=Nfouv THEN No_more | We've matched all measured stars.
6060     IF Kmax=5 THEN No_more | We have enough confirming stars.
6070     GOTO Next_star
6080 Next_j: NEXT J
6090 Next_star: NEXT N
6100 |
6110 No_more: IF Kmax>2 THEN SUBEXIT
6120 PRINT USING "/K"; "*** No additional stars found ** Assume false match"

```

```
6130 Kmax=0
6140 PRINT USING "/K"; "Replace new values of Euler parameters with old values and continue pairing."
6150 MAT Bvn=Sb ! Replace new Euler parameters with old values.
6160 GOTO Nextii ! Continue pairing--assume this orientation isn't correct.
6170 SUBEND
```

Perturb

This subroutine computes a time varying perturbation to Euler parameters. We have used simple sinusoidal variations added to the nominal values. Input data consist of the orbital frequency, time, the nominal values of the Euler parameters and the amplitude and frequency of the variations. The perturbed Euler parameters are returned to the calling program.

```

*****
*
*               P E R T U R B
*
*****

7810 SUB Perturb(SHRT Bnom(*),E(*),N(*),REAL Omega,Dt,B(*))
7820 OPTION BASE 1
7830   !
7840   B(1)=Bnom(1)+E(1)*COS(N(1)*Omega*Dt)
7850   B(2)=Bnom(2)+E(2)*SIN(N(2)*Omega*Dt)   ! Perturb the nominal Euler
7860   B(3)=Bnom(3)+E(3)*COS(N(3)*Omega*Dt)   ! parameters.
7870   B(4)=Bnom(4)+E(4)*SIN(N(4)*Omega*Dt)
7880   !
7890   Mag=SQR(DOT(B,E))   ! Normalize the new Euler parameters.
7900   MAT B=3/(Mag)
7910   !
7920 SUBEND

```

Phoeqn

This small subroutine uses the stellar colinearity equations to compute image coordinates. As input it needs the star direction cosines and the 3×3 rotation matrix. This routine returns the x and y coordinates normalized by lens focal length:

$$\frac{X}{f} = \frac{L_1 C_{11} + L_2 C_{12} + L_3 C_{13}}{L_1 C_{31} + L_2 C_{32} + L_3 C_{33}}$$

$$\frac{Y}{f} = \frac{L_1 C_{21} + L_2 C_{22} + L_3 C_{23}}{L_1 C_{31} + L_2 C_{32} + L_3 C_{33}}$$

where C_{ij} are elements of the rotation matrix and L_i are star direction cosines.

```

*****
*                                     *
*                               P H O E Q N                               *
*                                     *
*****

```

```

8170 ! Computes x,y coordinates for a particular star.
8180 SUB Phoeqn(L(*),C(*),Xpho,Ypho)
8190 OPTION BASE 1
8200 DIM Pho(3)
8210 !
8220 MAT Pho=C*_ ! Rotate direction cosines into new frame.
8230 Xpho=Pho(1)/Phc(3)
8240 Ypho=Pho(2)/Phc(3)
8250 !
8260 SUBEND

```

Post-mult

The function of this subroutine is to post-multiply, by a rotation matrix, a set of 4 matrices which are the partial derivatives, with respect to Euler parameters, of a second rotation matrix. For example, for field of view A we need the partial derivatives of the AN rotation matrix with respect to β_{BA} . Subroutine Mat-av computes the partials of AV with respect to β_{BA} . Then, since $AN = AV \cdot VN$

$$\frac{\partial AN}{\partial \beta_{BA}} = \frac{\partial AV}{\partial \beta_{BA}} \cdot VN$$

where the partials indicate derivatives of matrix elements.

```

11260 SUB Post_mult(C(*),Dc1(*),Dc2(*),Dc3(*),Dc4(*))
11270 OPTION BASE 1
11280 DIM T(3,3)
11290 !
11300 MAT T=Dc1*
11310 MAT Dc1=T
11320 MAT T=Dc2* !
11330 MAT Dc2=T ! Post-multiply derivative ma
11340 MAT T=Dc3* !
11350 MAT Dc3=T
11360 MAT T=Dc4*
11370 MAT Dc4=T
11380 !
11390 SUBEND

```

! Post-multiply derivative matrices by rotation matrix.

Pre-mult

This subroutine is similar to Post-mult. In this case, we pre-multiply partial derivative matrices by a rotation matrix. For example, since $AN = AV \cdot VN$,

$$\frac{\partial AN}{\partial \beta_{VN}} = AV \cdot \frac{\partial VN}{\partial \beta_{VN}}$$

where, it will be recalled, the partial derivatives are computed by Dircosb.

```
*****
*
*           P R E _ M U L T
*
*****
```

```
11040 SUB Pre_mult(C(*),Dc1(*),Dc2(*),Dc3(*),Dc4(*))
11050 OPTION BASE 1
11060 DIM T(3,3)
11070 !
11080 MAT T=C*Dc1
11090 MAT Dc1=T
11100 MAT T=C*Dc2
11110 MAT Dc2=T
11120 MAT T=C*Dc3 ! Pre-multiply derivative matrices by a rotation matrix.
11130 MAT Dc3=T
11140 MAT T=C*Dc4
11150 MAT Dc4=T
11160 !
11170 SUBEND
```

Proc-b

This subroutine controls the various functions of Process B. As input we need the Euler parameters describing the vehicle frame orientation, β_{VN} , and those describing the interlock between camera frames A and B, β_{BA} , along with the variance to associate with β_{BA} in the Kalman filter update. Usually, both sets of Euler parameters are updated by Process B before they are returned to the calling program. Process B also requires the coordinates of each measured star in FOV(A) and FOV(B) and returns the calculated coordinates for stars matched with measured stars.

The first step in this subroutine is to compute the rotation matrices BA from β_{BA} , from which we calculate AV, and matrix VN from β_{VN} . The unit vector for the boresight of FOV(A) is contained in the last row of matrix AN = AV · VN and is used by Access to retrieve a subcatalog of stars near this boresight. Pair-it is then called to match catalog and measured stars and to update β_{VN} . We then compute VN again and calculate BN = BA · AV · VN. The boresight unit vector, the last row of BN, is used by Access to again obtain a sub-catalog. Pair-it once again matches measured and catalog stars and updates β_{VN} .

There are several possible paths for Process B. If either FOV contains fewer than three stars, we skip any attempt to match stars in the FOV (we need at least three stars to confirm an orientation). Should FOV(A) and FOV(B) each contain fewer than three stars, we declare a failure condition for Process B and return to the calling program. In this case, no attempt is made to update β_{VN} or β_{BA} (and

no Kalman filter update is needed); the integrated values of β_{VN} and covariance matrix are used to start the analysis of the next Process A data set.

If only one FOV contains a sufficient number of stars, we call Least-1 and use up to 5 stars in that FOV to update β_{VN} (Pair-it updates β_{VN} using only 2 stars). Note that the interlock parameters, β_{BA} , are not updated; the same values are used on the subsequent data frame.

Usually there are a sufficient number of stars in both FOV(A) and FOV(B) (more than 2 in each) so we can correct both β_{VN} and β_{BA} . This is done by subroutine Least-2.

If β_{VN} has been updated, then we compute calculated image coordinates for all matched stars and return these to the calling program along with the 4×4 covariance matrix associated with β_{VN} .

```

*****
*
*           P R O C - B
*
*****

3020 SUB Proc_b(#2,Evn(*),Bba(*),Voc(*),Nfovma,Nfovmb,Ka,Kb,W,Sigxy,SHORT Xyma(
*),Xymb(*),Xyca(*),Xycb(*),REAL Cov8(*),Pba(*),Qba(*),Bbalsq(*))
3030 ! T. STRIKWERDA ..... 9 JUNE 1980.
3040 ! This subprogram is process B of Star Wars. This version uses Euler
3050 ! parameters and recovers interlock Euler parameters.
3060 OPTION BASE 1
3070 DIM Bore(3)
3080 DIM Vn(3,3),Av(3,3),Ba(3,3)
3090 DIM An(3,3),Bn(3,3),Bu(3,3),Tn(3,3)
3100 DIM Flda(5,5),Fldb(5,5),Fovm(10,4)
3110 DIM Kal(4,4),Lta(4,4),Bbae(4),T3(4),T4(4)
3120 DIM Cov(4,4)
3130 SHORT Fovst(100,4)
3140 COM Vg(3,3),INTEGER Table(529,2)
3150 RAD
3160 REDIM Cov3(8,8)
3170 !
3180 F=70 ! Some constants.
3190 Fe=2.42536
3200 Fl=F+Fe
3210 Tol=9.25E-6
3220 Radius=5.7*PI/180
3230 Sigma=1*PI/1E0
3240 !
3250 ! Calculate interlock matrices BA and AV.
3260 !
3270 CALL Dircost(Bta(*),Ba(*),0,Tn(*),Tn(*),Tn(*),Tn(*))
3280 CALL Mat_av(Ba(*),Tn(*),Tn(*),Tn(*),Tn(*),Av(*),0,Tn(*),Tn(*),Tn(*),Tn(*))
3290 Ka=0
3300 Kb=0
3310 !
3320 PRINT USING "/K";" Start for FOV(A)"
3330 REDIM Fovm(Nfovma,4)
3340 IF Nfovma<3 THEN Fov_b
3350 FOR I=1 TO Nfovma
3360 Fovm(I,4)=Xyma(I,3) ! Normalize image coord. by focal length.
3370 Fovm(I,1)=Xyma(I,1)/Fl
3380 Fovm(I,2)=Xyma(I,2)/Fl
3390 Fovm(I,3)=1/SQR(Fovm(I,1)^2+Fovm(I,2)^2+1)
3400 NEXT I
3410 CALL Dircost(Bun(*),Vn(*),0,Tn(*),Tn(*),Tn(*),Tn(*))
3420 MAT An=Av*Vn
3430 FOR I=1 TO 3
3440 Bore(I)=An(3,I) ! Boresight unit vector for FOV(A).
3450 NEXT I
3460 PRINT USING Form5;"Boresight direction cosines for FOV(A):",Bore(*)
3470 Form5: IMAGE /K/3(MD.DDDDDD,X)

```

```

3480 CALL Access(#2,Nfova,Bore(*),Sigma,Radius,Voc(*),Foust(*))
3490 CALL Pair_it(Fcvt(*),Nfova,Fovm(*),Nfovma,Bvn(*),Av(*),Tol,Sigxy,F1,W,Fld
a(*),Ka)
3500 Ka2=Ka+Ka
3510 BEEP ! Done with FOV(A).
3520 !
3530 Fov_b: !
3540 PRINT USING "/k";" Start for FOV(B)"
3550 IF Nfovmb<3 THEN Options
3560 !
3570 CALL Dircost(Bvn(*),Vn(*),0,Tn(*),Tn(*),Tn(*),Tn(*))
3580 CALL Dircost(Bta(*),Ba(*),0,Tn(*),Tn(*),Tn(*),Tn(*))
3590 MAT Bv=Ba*Vn
3600 MAT Bn=Bv*Vn
3610 FOR I=1 TO 3
3620 Bore(I)=Br(3,I) ! Boresight unit vector for FOV(B).
3630 NEXT I
3640 PRINT USING Form5;"Boresight direction cosines for FOV(B):",Bore(*)
3650 REDIM Fovm(Nfovmb,4)
3660 !
3670 FOR I=1 TO Nfovmb
3680 Fovm(I,4)=Xymb(I,3) ! Normalize image coord. by focal length.
3690 Fovm(I,1)=Xymb(I,1)/F1
3700 Fovm(I,2)=Xymb(I,2)/F1
3710 Fovm(I,3)=1/SQR(Fovm(I,1)^2+Fovm(I,2)^2+1)
3720 NEXT I
3730 !
3740 CALL Access(#2,Nfovmb,Bore(*),Sigma,Radius,Voc(*),Foust(*))
3750 CALL Pair_it(Fcvt(*),Nfovmb,Fovm(*),Nfovmb,Bvn(*),Bu(*),Tol,Sigxy,F1,W,Fld
b(*),Kb)
3760 Kb2=Kb+Kb
3770 BEEP
3780 !
3790 Options: !
3800 IF (Ka>2) AND (Kb>2) THEN Combine ! Do least-squares for two FOV.
3810 IF (Ka<3) AND (Kb<3) THEN Failed ! PUNT!!!
3820 REDIM Cov8(4,4)
3830 IF (Ka>2) AND (Kb<3) THEN CALL Least_1(Bvn(*),Av(*),Flda(*),Ka,W,Cov8(*),C
onverge)
3840 IF (Ka<3) AND (Kb>2) THEN CALL Least_1(Bvn(*),Bu(*),Fldb(*),Kb,W,Cov8(*),C
onverge)
3850 !
3860 GOTO Save_results
3870 !
3880 Combine: !
3890 !
3900 MAT Bbae=Bba ! Save estimated interlock parameters.
3910 CALL Least_2(Bvn(*),Bba(*),Ka,Flda(*),Kb,Fldb(*),W,Cov8(*))
3920 MAT Bbalsq=Bba
3930 ! Perform Kalman filter update for interlock parameters.
3940 FOR I=1 TO 4
3950 FOR J=1 TO 4
3960 Lba(I,J)=Cov8(4+I,4+J)
3970 NEXT J
3980 NEXT I

```

```

3990      !
4000      MAT Pba=Pba+Cba      ! Get covariance matrix at this time.
4010      MAT Kal=_ba+Fba
4020      MAT Lba=INV(Kal)
4030      MAT Kal=Pba*Lba      ! Kalman gain matrix.
4040      MAT T3=Bba-Btae
4050      MAT T4=Kal*T3      ! Corrections to interlock parameters.
4060      MAT Bba=Btae+T4
4070      MAT Lba=Kal*Fba      ! Corrections to covariance matrix.
4080      MAT Pba=Pba-Lba
4090      !
4100      Save_results:      !
4110      !
4120      CALL DirCosb(Bun(*),Vn(*),0,Tn(*),Tn(*),Tn(*),Tn(*))
4130      CALL DirCosb(Bba(*),Ba(*),0,Tn(*),Tn(*),Tn(*),Tn(*))
4140      CALL Mat_av(Ea(*),Tn(*),Tn(*),Tn(*),Tn(*),Av(*),0,Tn(*),Tn(*),Tn(*),Tn(*))
4150      MAT An=Au+Vn
4160      MAT Xyca=ZER
4170      IF Ka=0 THEN Save_b
4180      !
4190      FOR K=1 TO Ka      ! Calculate image coord. for each
4200          FOR I=1 TO 3      ! matched star.
4210              Bore(I)=F1da(K,I)
4220          NEXT I
4230          CALL Ptoeqr(Bore(*),An(*),X,Y)
4240          Xyca(K,1)=X
4250          Xyca(K,2)=Y
4260      NEXT K
4270      !
4280      MAT Xyca=Xyca*(F1)
4290      Save_b:      !
4300      MAT Bn=Ba+An
4310      MAT Xycb=ZER
4320      IF Kb=0 THEN End
4330      !
4340      FOR K=1 TO Kb
4350          FOR I=1 TO 3
4360              Bore(I)=F1db(K,I)
4370          NEXT I
4380          CALL Ptoeqr(Bore(*),Bn(*),X,Y)
4390          Xycb(K,1)=X
4400          Xycb(K,2)=Y
4410      NEXT K
4420      MAT Xycb=Xycb*(F1)
4430      !
4440      Form1: IMAGE K/4(MD.DDDDDDE,X)/
4450      Form4: IMAGE K,X,DD/10(2(MD.DDDD,X)/)
4460      End:      !
4470      SUBEXIT
4480      !
4490      Failed: PRINT "***** PROCESS B FAILED *****"
4500              PRINT "Fewer than 3 stars in each FOV. No attempt to perform "
4510              PRINT "least-squares correction. Use old values for orientation."
4520              GOTO Save_results
4530      SUBEND

```

Runge

The subroutine integrates both the state differential equations and the matrix Riccati equation, using Runge-Kutta methods. As discussed in Appendix 8, we partition the Riccati equation into four parts and only two of these need to be integrated numerically.

Both of the equations are integrated with two-cycle Runge-Kutta methods. However, since the covariance matrix should be relatively constant in steady-state, we use a step size, for the Riccati equation, equal to the time between data frames (currently, 30 seconds). The step size for the state integration is much smaller (currently, 0.5 sec or 60 steps between frames).

The first task in this subroutine is to partition the covariance matrix and evaluate the right-hand-side of the Riccati equation at the start of the time interval. We then integrate the state equation, through repeated use of two-cycle Runge-Kutta methods, until we reach the end of the interval. The right-hand-side of the Riccati equation is again evaluated, this time at the end of the interval, and the integrated covariance matrix calculated.

```

*****
*                                     *
*                               RUNGE *
*                                     *
*****

13600 SUB Runge(Tk,Delt,Step,SHORT W1(*),W2(*),W3(*),REAL Xk(*),P(*),Q(*),Sigb)
13610 OPTION BASE 1
13620 DIM X(4),P11(4,4),P21(3,4),P11p(4,4),P21p(3,4),P22(3,3)
13630 DIM D1(4),L1(4,4),L1(3,4),Sumk(4,4),Sum1(3,4)
13640 DIM S1(4),A11(4,4),A12(4,3),W(3),B(3)
13650 DIM Q22(3,3),P22p(3,3)
13660 !
13670 MAT Q22=ID4
13680 MAT Q22=Q22+((Sigb/30)^2) ! Factor of 30 may be changed to tune Kalman
13690 ! filter for bias recover.
13700 DISP "Runge"
13710 !
13720 T=Tk
13730 FOR I=1 TO 4
13740   FOR J=1 TO 4
13750     P11(I,J)=P(I,J) ! Get upper left 4x4 portion of cov. matrix.
13760   NEXT J
13770 NEXT I
13780 !
13790 FOR I=1 TO 3
13800   FOR J=1 TO 4
13810     P21(I,J)=P(4+I,J) ! Get lower left 3x4 portion of cov. matrix.
13820   NEXT J
13830 NEXT I
13840 !
13850 FOR I=1 TO 3
13860   B(I)=Xk(4+I) ! Get current bias values.
13870   FOR J=1 TO 3
13880     P22(I,J)=P(4+I,4+J) ! Get lower right 3x3 portion of cov. matrix.
13890   NEXT J
13900 NEXT I
13910 !
13920 REDIM X(4),Xk(4)
13930 MAT X=Xk
13940 MAT P11p=P11
13950 MAT P21p=P21
13960 MAT P22p=P22
13970 ! Compute time derivative of covariance matrix at time t(initial).
13980 W(1)=W1(1)
13990 W(2)=W2(1) ! Use gyro rates from beginning of interval.
14000 W(3)=W3(1)
14010 !
14020 CALL Mata(1,W(*),Xk(*),A11(*),A12(*))
14030 CALL Deriv(P11p(*),P21p(*),P22p(*),Q(*),A11(*),A12(*),K1(*),L1(*))
14040 !
14050 MAT Sumk=K1
14060 MAT Sum1=L1

```

```

14070 MAT A11=K1*(Step)
14080 MAT P11p=P11+A11
14090 MAT A11=L1*(Step)
14100 MAT P21p=P21+A11
14110 REDIM A11(4,4)
14120 MAT P22p=Q22*(Step)
14130 MAT P22p=P22+P22p
14140 !
14150 ! Begin state integration.
14160 !
14170 MAT W=W-B ! Subtract bias values.
14180 CALL Mata(0,W(*),X(*),A11(*),A12(*))
14190 !
14200 FOR It=2 TO Step/Delt+1 ! This is a series of 2 - step Runge-Kutta
14210 MAT D1=A11*Xk ! integrations.
14220 MAT S1=(Delt)*D1
14230 MAT X=Xk+S1
14240 W(1)=W1(It)
14250 W(2)=W2(It) ! Get gyro rates for interval.
14260 W(3)=W3(It) ! Note: These are measured rates in gyro frame.
14270 MAT N=W-B ! Subtract the biases.
14280 CALL Mata(0,W(*),X(*),A11(*),A12(*))
14290 MAT S1=A11*X
14300 MAT X=D1+S1
14310 MAT X=(.5*Delt)*X
14320 MAT Xk=X+X
14330 DISP Xk(*);
14340 Tk=Tk+Delt
14350 NEXT I;
14360 !
14370 Mag=SQR(DOT(Xk,Xk)) ! Normalize Euler parameters.
14380 MAT Xk=Xk/(Mag)
14390 ! Compute time derivative of covariance matrix at time t(final).
14400 W(1)=W1(61)
14410 W(2)=W2(61)
14420 W(3)=W3(61)
14430 !
14440 CALL Mata(1,W(*),Xk(*),A11(*),A12(*))
14450 CALL Deriv(P11p(*),P21p(*),P22p(*),Q(*),A11(*),A12(*),K1(*),L1(*))
14460 !
14470 ! Compute updated covariance matrix.
14480 !
14490 MAT Sunk=Sunk+K1 ! Compute upper-left 4x4 matrix.
14500 MAT Sunk=Sunk*(Step/2)
14510 MAT P11=P11+Sunk
14520 MAT Sum1=Sum1+L1 ! Compute lower-left 3x4 matrix.
14530 MAT Sum1=Sum1*(Step/2)
14540 MAT P21=P21+Sum1
14550 MAT P22p=Q22*(Step)
14560 MAT P22=P22+P22p
14570 ! Fill the upper left 4x4 of the covariance matrix.
14580 FOR I=1 TO 4
14590 FOR J=1 TO 4
14600 P(I,J)=P11(I,J)
14610 NEXT J

```

```
14620 NEXT I
14630      ! Fill the lower left 3x4 part of covariance matrix
14640      ! and the upper 4x3 part with the transpose.
14650 FOR I=1 TO 3
14660   FOR J=1 TO 4
14670     P(4+I,J)=P21(I,J)
14680     P(J,4+I)=P21(I,J)
14690   NEXT J
14700 NEXT I
14710 FOR I=1 TO 3
14720   FOR J=1 TO 3
14730     P(4+I,4+J)=P22(I,J)
14740   NEXT J
14750 NEXT I
14760      !
14770 REDIM KK(7)
14780      !
14790 DISP
14800 SUBEND
```

Secant

Subroutine Secant uses the secant method to update the partial derivative matrix used for least-squares correction (containing the derivatives of image coordinates with respect to the Euler parameters, β). If we let $X = X(\beta)$ be the set of function (colinearity equations) which produce image coordinates for stars as a function of Euler parameters, then at the k th iteration we have β^k , the coordinates X^k , and the partial derivative matrix $A^k = \partial X / \partial \beta|_k$ (determined by Dxdbeta). By least-squares we obtain corrections to β^k to get $\beta^{k+1} = \beta^k + \Delta\beta^k$. These are used to compute new coordinate X^{k+1} so the changes are

$$\delta X^k = X^{k+1} - X^k.$$

However, the linearly predicted changes in X are

$$\Delta X^k = A^k \Delta\beta^k.$$

We proceed to modify the derivative matrix (be adding corrections) so that the linearly predicted changes will agree with the actual changes:

$$\delta X^k = (A^k + C^k) \Delta\beta^k.$$

No unique solution to this equation exists so we introduce

$$\phi = \sum_{ij} (C_{ij}^k)^2 \text{ and minimize this criterion subject to}$$

$$\delta X^k - A^k \Delta\beta^k - C^k \Delta\beta^k = 0.$$

Using the Lagrange multiplier technique we minimize

$$\Phi = \sum_i \sum_j (C_{ij}^k)^2 + \lambda^T (\delta X^k - A^k \Delta\beta^k - C^k \Delta\beta^k).$$

The necessary conditions require

$$\partial \Phi / \partial C_{ij}^k = 0$$

and

$$\partial\Phi/\partial\lambda_i = 0$$

or

$$C_{ij}^k = \frac{1}{2} \lambda_i \Delta\beta_j^k$$

and

$$\delta X^k - A^k \Delta\beta^k - C^k \Delta\beta^k = 0.$$

Matrix C^k can be expressed as the outer product of two vectors:

$$C^k = \frac{1}{2} \lambda (\Delta\beta^k)^T.$$

We can now substitute for matrix C^k in the second necessary condition and solve this equation for the Lagrange multipliers:

$$\lambda = 2(\delta X^k - A^k \Delta\beta^k) / (\Delta\beta^k)^T \Delta\beta^k$$

and substitute this for λ in the first necessary condition. Thus, the updated partial derivative matrix is

$$A^{k+1} = A^k + (\delta X^k - A^k \Delta\beta^k) (\Delta\beta^k)^T / (\Delta\beta^k)^T \Delta\beta^k.$$

The secant method works best when we are near the solution vector, $\Delta\beta$. As a check on the performance of this method, we compute the root-mean-square difference between calculated and measured coordinates at each iteration. If this parameter ever increases from its last value, we start over with exact derivatives (computed by Dxdbeta) before continuing with the least-square correction.

```

*****
*
*           S E C A N T
*
*****

7850  ! Secant method of derivative update.
7860  SUB Secant(H(*),Delx(*),Ddy(*),Idim,Jdim)
7870  OPTION BASE 1
7880  DIM T1(1,Jdim),T2(Idim,1),T3(Idim,Jdim),Sumat(Idim)
7890  !
7900  DISP "Secant"
7910  !
7920  Sumt2=DOT(Delx,Delx)
7930  MAT Sumat=R+Delx      ! Predicted changes in coordinates.
7940  MAT Sumat=Ddy-Sumat  ! Difference between actual and predicted
7950  MAT Sumat=Sumat/(Sumt2) ! changes.
7960  !
7970  FOR I=1 TO Idim      ! Fill vectors for outer product.
7980    T2(I,1)=Sumat(I)   ! (see Appendix of Final Report.)
7990  NEXT I
8000  FOR J=1 TO Jdim
8010    T1(1,J)=Ddelx(J)
8020  NEXT J
8030  !
8040  MAT T3=T2*T1      ! Compute corrections to derivatives.
8050  MAT R=R+T3        ! Add corrections.
8060  !
8070  DISP
8080  SUBEND

```

Sort

This subroutine sorts an array and a column vector according to the values in the column vector. The order of sorting is from the largest to the smallest values and the array and vector must have the same number of rows. We use a simple "bubble" sort method - make repeated searches through the list, each time bringing the next largest value to the next available location in the list. In this version, in order to save computer time, we use a vector to save the re-ordering sequence of the column vector and use this sequence to re-order the array as the last step.

```

*****
*
*               S O R T
*
*****

9730 ! This subroutine sorts arrays by rows, given a parameter vector of
9740 ! same dimension as the number of rows.
9750 SUB Sort(A(*),SHORT B(*),REAL N,M)
9760 OPTION BASE 1
9770 DIM C(N),S(N,M)
9780 !
9790 DISP "Sort"
9800 !
9810 FOR K=1 TO N
9820   C(K)=K           ! Fill index vector.
9830 NEXT K
9840 N1=N-1
9850 !
9860 FOR K=1 TO N1      ! Bubble sort -- bring largest value to top.
9870   Jj=K
9880   Test=A(K)
9890   K1=K+1
9900   FOR I=K1 TO N
9910     IF Test>A(I) THEN Continue  ! Search for largest value.
9920     Jj=I
9930     Test=A(I)
9940 Continue: NEXT I
9950   IF Jj=K THEN Continue2
9960   A(Jj)=A(K)
9970   A(K)=Test        ! Place next largest value in next location.
9980   T=C(K)
9990   C(K)=C(Jj)       ! Switch indices as well.
10000  C(Jj)=T
10010 Continue2: NEXT K
10020 !
10030 FOR I=1 TO N
10040   K=C(I)
10050   FOR L=1 TO M
10060     S(I,L)=B(K,L)  ! Reorder array in same order--refer to index vector.
10070   NEXT L
10080 NEXT I
10090 !
10100 MAT B=S
10110 !
10120 DISP
10130 SUBEND

```

AD-A103 806

VIRGINIA POLYTECHNIC INST AND STATE UNIV BLACKSBURG --ETC F/G 17/7
STAR PATTERN RECOGNITION AND SPACECRAFT ATTITUDE DETERMINATION.(U)
MAY 81 T E STRIKWERDA, J L JUNKINS

DAAK70-78-C-0038

UNCLASSIFIED

ETL-0260

NL

3 OF 3

ALL A
10-8-81



--	--	--	--	--	--	--	--	--

END
DATE FILMED
10-8-81
DTIC

Sample output from data generator program (DATGEN):
Program setup, data for frame 2 and beginning of frame 3.

***** PROGRAM DATGEN *****

Do you want to use realistic gyro rate history (Y/N)? Y

Place disk with gyro rates (Filename: 'Wtrue') in :F8,1.....then push CONT.

Place star catalog disk (Filenames: 'Tab22' and 'Miss220') in :F8,1...
...then push CONT.

Has Table of star catalog cell positions been read-in?(Y/N) N

File name for simulation run ('Simnnn:F8,1'...where nnn is 3 num.):

Sim042:F8,1

Period : 90.03 minutes

Do you want variations in Euler para. relating V frame to Gyro frame (Y/N)? Y

Do you want variations in Euler para. relating B frame to A frame (Y/N)? Y

Do you want time varying gyro biases (Y/N)? Y

Do you want to add noise to image coordinates(Y/N)?Y

Do you want noise added to rate gyro data (Y/N)?Y

Position (km) and velocity (km/sec) of Satellite:

6.6526E+03 0.0000E+00 0.0000E+00
0.0000E+00 2.0004E+00 7.4760E+00

Matrix GN:

0.000000 .258319 .965926
0.000000 -.965926 .258319
1.000000 0.000000 0.000000

Bug...nominal Euler Parameters between frames V-G:

1.000000 0.000000 0.000000 0.000000

Bug...true Euler Parameters between frames V-G:

1.000000 0.000000 .000015 0.000000

Bvn...Initial Euler Parameters between frames V-N:

.092295 .701047 .092297 .701068

***** FRAME : 2 *****

Matrix VG:

1.000000 0.000000 -.000029
0.000000 1.000000 0.000000
.000029 0.000000 1.000000

Satellite time from start of simulation: 30.00 seconds

Bvn...True Euler Parameters between frames V-N:

.090640 .688586 .093727 .713240

Bba...Nominal Euler Parameters between frames B-A:

.707107 .707107 0.000000 0.000000

Bba...True Euler Parameters between frames B-A:

.707131 .707383 .000097 0.000000

Matrix BA :

1.000000	.000137	-.000137
.000137	.000369	1.000000
.000137	-1.000369	.000069

Position (km) and velocity (km/sec) of Satellite:

6.6485E+03	6.0390E+01	2.2426E+02
-2.7014E-01	2.0322E+00	7.4723E+00

Position (km) and velocity (km/sec) of Earth:

-1.4960E+08	8.1343E+02	-3.5630E+02
1.7790E-04	2.7314E+01	-1.1877E+01

Total velocity of satellite (km/sec):

-2.6997E-01	2.9317E+01	-4.4843E+00
-------------	------------	-------------

Matrix AN:

.999384	.008786	.033971
.025019	.500383	-.865442
-.024602	.865759	.499855

Bore sight unit vector:

-.024602	.865759	.499855
----------	---------	---------

Polar angle: 63.01

Longitude angle: 91.63

Indices for four cells:

16	16	14	14
8	9	7	8

Number of stars from the catalog: 14.00

Number of stars in this FOV: 10.00

True image coordinates (mm):

5.128090	2.681500	4.503000
4.462200	3.126360	4.509000
2.580310	2.440960	4.810000
.891715	.633460	3.244000
-1.584060	.529351	2.990000
5.024780	-2.588570	3.085000
-.543383	-3.457330	4.166000
-.950058	-4.177740	4.574000
-4.615360	-3.573780	4.966000
-4.840800	-1.151580	4.936000

Measured image coordinates (mm):

5.125990	2.680390	4.523250
4.458730	3.124930	4.560540
2.581860	2.440350	4.862560
.896847	.637305	3.259580
-1.582270	.524518	2.987110
5.018640	-2.587440	2.961420
-.547052	-3.453350	4.174510
-.953583	-4.175710	4.633090
-4.610930	-3.573390	4.875600
-4.840900	-1.151770	4.929290

Matrix AN:

.999391	.008736	.033784
-.024464	.865795	.499801
-.024884	-.500323	.865481

Bore sight unit vector:

-.024884	-.500323	.865481
----------	----------	---------

Polar angle: 33.06

Longitude angle: 267.15

Indices for four cells:

188

8	8	5	6
13	12	10	9

Number of stars from the catalog: 9.00

Number of stars in this FOV: 5.00

True image coordinates (mm):

.438158	-3.896320	2.313000
2.604860	-3.363190	4.900000
5.575850	-1.474120	4.835000
-3.296900	-2.229360	4.825000
-1.927940	2.696350	4.342000
0.000000	0.000300	0.000000
0.000000	0.000300	0.000000
0.000000	0.000300	0.000000
0.000000	0.000300	0.000000
0.000000	0.000300	0.000000

Measured image coordinates (mm):

.437323	-3.897300	2.321970
2.603590	-3.365270	4.866520
5.579470	-1.471220	4.821620
-3.305680	-2.229210	4.858870
-1.928110	2.694140	4.363480
0.000000	0.000300	0.000000
0.000000	0.000300	0.000000
0.000000	0.000300	0.000000
0.000000	0.000300	0.000000
0.000000	0.000300	0.000000

Biases...true values:

-0.000007	.000310	-.000015
-----------	---------	----------

Frame: 2 Number of stars: 10 5

***** FRAME : 3 *****

Matrix VG:

1.000000	.000308	-.000029
-.000008	1.000300	-.000003
.000029	.000303	1.000000

Satellite time from start of simulation: 60.00 seconds

Bun...True Euler Parameters between frames V-N:

.089034	.676167	.095378	.725102
---------	---------	---------	---------

Bba...Nominal Euler Parameters between frames B-A:

.707107	.707107	0.000000	0.000000
---------	---------	----------	----------

Bba...True Euler Parameters between frames B-A:

.707139	.707375	.000095	.000003
---------	---------	---------	---------

Matrix BA :

1.000000	.000140	-.000130
.000130	.000390	1.000000
.000140	-1.000300	.000090

Position (km) and velocity (km/sec) of Satellite:

6.6364E+03	1.2311E+02	4.4825E+02
-5.3996E-01	1.9905E+00	7.4586E+00

Position (km) and velocity (km/sec) of Earth:

-1.4960E+00	1.6389E+03	-7.1260E+02
3.5581E-04	2.7314E+01	-1.1877E+01

Total velocity of satellite (km/sec):

-5.3960E-01	2.9313E+01	-4.4180E+00
-------------	------------	-------------

Sample output from data analysis program (COMBIN):
Program setup and beginning of frame 2.

*****PROCESS B AND C*****

Doing Proc B (Y/N)? Y

Doing Proc C (Y/N)? Y

Insert star catalog disk into F8,1....Then press CONT

Has cell table been read-in? N

Input file name and device with simulation data: Sim042:F8,1

Satellite orbit major axis (km): 6653
Earth orbit major axis (km): 1.4960E+08
Satellite orbit inclination (deg.): 75
Rate gyro data spacing (sec): .50
Runge-Kutta time step (sec): 30.00
Gyro standard deviation (rad/sec): 4.848E-06

Input weight in arcseconds for interlock variance (2,5,etc.) 5.000

Input Gyro Bias Standard Deviation (Degrees/Hr) .50

Q Matrix:

2.350E-11	0.000E+00	0.000E+00
0.000E+00	2.350E-11	0.000E+00
0.000E+00	0.000E+00	2.350E-11

Do you want to offset matrix VG (V-frame to Gyro frame) N

Matrix VG for this run:

1.000E+00	0.000E+00	0.000E+00
0.000E+00	1.000E+00	0.000E+00
0.000E+00	0.000E+00	1.000E+00

***** RECORD NUMBER: 2 *****

Bvn....True Euler parameters between V and N frames:
.090671 .683716 .093893 .713186

Bba....True Euler parameters between B and A frames:
.707131 .707083 .000097 0.000000

Bba....Current Euler parameters between B and A frames:
.707131 .707083 .000097 0.000000

Components of total velocity (km/sec):
-2.700E-01 2.932E+01 -4.404E+00

Number of stars in each FOV:
FOV(A): 10
FOV(B): 5

Bvn....Integrated Euler parameters between V and N frames
and gyro biases:
.089241 .676321 .095545 .725192
0.000E+00 0.000E+00 0.000E+00

Components of total velocity (km/sec):
-2.700E-01 2.932E+01 -4.404E+00

Sample output from data analysis program (COMBIN):
Analysis of data from frame 5.

***** RECORD NUMBER: 5 *****

Bun....True Euler parameters between V and N frames:
.085633 .653446 .098509 .748253

Bba....True Euler parameters between B and A frames:
.707153 .707061 .000084 .000010

Bba....Current Euler parameters between B and A frames:
.707146 .707067 .000099 -.000000

Components of total velocity (km/sec):
-1.077E+00 2.930E+01 -4.473E+00

Number of stars in each FOV:
FOV(A): 8
FOV(B): 6

Bun....Integrated Euler parameters between V and N frames
and gyro biases:
.085706 .650376 .098562 .748299
-5.633E-06 4.482E-06 -1.359E-05

Components of total velocity (km/sec):
-1.077E+00 2.930E+01 -4.473E+00

Start for FOV(A)

Boresight direction cosines for FOV(A):
-.098483 .864305 .493232

Polar angle: 60.4 Degrees
Longitude angle: 96.5 Degrees

Cell indices:
16 16 14 14
9 8 8 7

Table of Cos(Theta) for Measured Stars.

Star	Star	Cos(Theta)
1	2	.994825
1	3	.993700
1	4	.997851
2	3	.999822
1	5	.997526
2	4	.990894
1	6	.998723
2	5	.995755
3	4	.990641
1	7	.998603
2	6	.997739
3	5	.995960
1	8	.998383
2	7	.998021
3	6	.997476
4	5	.998695
2	8	.998996

3	7	.997766
4	6	.997697
3	8	.990533
4	7	.997395
5	6	.999438
5	7	.999340
5	8	.998429
6	8	.997745
7	8	.997445

Begin Pairing Catalog Stars and Comparing To Measured Pairs.

Number of stars from catalog: 15

Star	Star	Cos(Theta)
1	2	.999990
1	3	.998015

>>>> Catalog Pair Matched with Measured Pair <<<<

Measured pair:

Star	Star	Cos(Theta)
7	2	.998021

Least-Squares Correction For One FOV

RMS error for normalized image coordinates: 1.6748E-04

Beta(old)	Beta(new)	Delta(Beta)
.0857057	.0852774	-.0004283
.6503763	.6509618	.0005855
.0985616	.0989390	.0003773
.7482986	.7477891	-.0005095

RMS change in Euler parameters: 4.8172E-04

RMS error for normalized image coordinates: 2.3134E-05

Beta(old)	Beta(new)	Delta(Beta)
.0852774	.0852771	-.0000003
.6509618	.6509621	.0000003
.0989390	.0989394	.0000004
.7477891	.7477882	-.0000009

RMS change in Euler parameters: 5.4913E-07

Number of iterations: 2 Converge=1

<u>Test For Additional Stars</u>					
x(calc)	x(meas)	y(calc)	y(meas)	Dx	Dy
.3588	.3567	-1.1708	-1.1696	.0021	.0012
.6301	.6335	-1.3228	-1.3184	.0034	.0044
-3.6080	-3.6060	1.0902	1.0890	.0021	.0012
3.7541	3.7560	.5865	.5992	.0019	.0128
1.0280	1.0338	-3.7184	-3.7160	.0058	.0024

Start for FOV(B)

Boresight direction cosines for FOV(B):

-.096652 -.501563 .859646

Polar angle: 30.7 Degrees

Longitude angle: 259.1 Degrees

Cell indices:

8	8	6	6
12	13	9	10

Table of Cos(Theta) for Measured Stars.

Star	Star	Cos(Theta)
1	2	.995801
1	3	.998750
1	4	.991635
2	3	.997596
1	5	.992973
2	4	.997046
1	6	.993075
2	5	.992756
3	4	.996696
2	6	.991646
3	5	.997282
3	6	.990925
4	5	.997517
4	6	.998411
5	6	.995345

Begin Pairing Catalog Stars and Comparing To Measured Pairs.

Number of stars from catalog: 16

Star	Star	Cos(Theta)
1	2	.996682
1	3	.997515

>>>> Catalog Pair Matched with Measured Pair <<<<

Measured pair:

Star	Star	Cos(Theta)
5	4	.997517

Least-Squares Correction For One FOV

RMS error for normalized image coordinates: 1.2863E-03

Beta(old)	Beta(new)	Delta(Beta)
.0852771	.0853085	.0000315
.6509621	.6501618	-.0008003
.0989394	.0988372	-.0001022
.7477882	.7464948	.0007066

RMS change in Euler parameters: 5.3647E-04

RMS error for normalized image coordinates: 9.5265E-06

Beta(old)	Beta(new)	Delta(Beta)
.0853085	.0853081	-.0000004
.6501618	.6501616	-.0000002
.0988372	.0988367	-.0000004
.7464948	.7464943	-.0000005

RMS change in Euler parameters: 3.9327E-07

Number of iterations: 2 Converge=1

Test For Additional Stars

x(calc)	x(meas)	y(calc)	y(meas)	Dx	Dy
1.8908	1.8912	-2.1949	-2.1940	.0003	.0009
-1.7222	-1.7192	2.4753	2.4632	.0031	.0121
3.6270	3.6267	2.6151	2.6141	.0003	.0009
-3.6754	-3.6810	-2.1516	-2.1678	.0056	.0162
-4.7927	-4.7999	4.4085	4.3959	.0071	.0126

Least-Squares Correction For Two FOV

Correct orientation using 5 stars from FOV(A) and 5 stars from FOV(B).

RMS error in normalized image coordinates: 5.7645E-04

Euler parameters and corrections:

B(V-N)	delta-B(V-N)	B(B-A)	delta-B(B-A)
.0856257	.0303175	.7071793	.0000330
.6504280	.0302664	.7070343	-.0000329
.0985211	-.0303156	-.0000228	-.0001221
.7482684	-.0302259	-.0001544	-.0001542

RMS change in Euler parameters: 2.1310E-04

RMS error in normalized image coordinates: 4.8614E-05

Euler parameters and corrections:

B(V-N)	delta-B(V-N)	B(B-A)	delta-B(B-A)
.0856257	.0300001	.7071792	-.0000002
.6504279	-.0300001	.7070344	.0000002
.0985212	.0300001	-.0000227	.0000000
.7482683	-.0300001	-.0001544	.0000000

RMS change in Euler parameters: 1.0901E-07

RMS error in normalized image coordinates: 4.8614E-05

Euler parameters and corrections:

B(V-N)	delta-B(V-N)	B(B-A)	delta-B(B-A)
.0856257	.0300000	.7071792	-.0000000
.6504279	.0300000	.7070344	.0000000
.0985212	.0300000	-.0000227	.0000000
.7482683	-.0300000	-.0001544	.0000000

RMS change in Euler parameters: 7.6322E-09
(End of least-squares for two FOV)

Kalman Filter State Estimation

True	State Vectors			Differences		
	Integ.	Proc. B	Opt. Est.	(I-T)	(B-T)	(O-T)
8.563E-02	8.571E-02	8.563E-02	8.564E-02	7.280E-05	-7.171E-06	5.125E-06
6.504E-01	6.504E-01	6.504E-01	6.504E-01	-7.019E-05	-1.861E-05	-2.872E-05
9.851E-02	9.855E-02	9.852E-02	9.853E-02	5.228E-05	1.184E-05	1.641E-05
7.483E-01	7.483E-01	7.483E-01	7.483E-01	4.579E-05	1.544E-05	2.231E-05
-7.062E-06	-5.633E-06	-5.633E-06	-6.234E-06	1.430E-06	1.430E-06	8.284E-07
7.796E-06	4.482E-06	4.482E-06	5.567E-06	-3.314E-06	-3.314E-06	-2.230E-06
-1.725E-05	-1.359E-05	-1.359E-05	-1.556E-05	3.656E-06	3.656E-06	1.685E-06

Norm of optimal estimate - 1: .00000007

(end of frame)

ND